CIS399: The Science of Data Ethics **Lecture 18: March 26, 2019**
Instructor: Michael Kearns & Ani Nenkova Scribes: Sam Holland, Thomas Kumpf

1 Recap

First, let's recap our discussion on privacy and security issues from our last meeting. We discussed the privacy and security issues associated with databases of human information. Each row contains information (features) about each individual, which is fed into an algorithm that predicts, for example, the probability that you have some disease. The output could then be some sort of useful statistic or a neural network.

When discussing privacy, it's conceptually helpful to distinguish the difference between the security of the database and the output. Security of the database consists of making sure only allowed parties are able to see it (cryptography), while security of the output is a bit different, because these results may be released to the general public – especially if they shed light on what features can cause a certain type of disease. The danger here is when looking at this public output, it's possible to derive unwanted inferences about specific personal information in the database.

This example brings up topics of cryptography, network security, and data security, all of which are big topics in computer science. Importantly, cryptography actually predates computers — people have been sending encoded messages forever, although with the creation of computers it became far more important and many algorithms were developed in the last 50 years. When talking about these social norms like privacy, security, fairness, it's important to think about definitions, what we want to ask for, what we don't want to ask for, so that we can design algorithms that meet these conditions.

2 Cryptography

2.1 One-time Pad

The one-time pad is an encryption technique originally proposed in the 1950s by Claude Shannon, often referred to as the inventor of information theory. This method produces a nice "upper bound" definition of privacy, so what you see as the output is actually indistinguishable from a random string. In other words, the output looks exactly like a coin flip. The procedure is described below:

`One-Time Pad`
One party has some message to send: $a = \{a_1, a_2, ..., a_n\}$, $a_i \in \{0, 1\}$
Both parties privately decide on a random sequence of bits: $b = \{b_1, b_2, ..., b_n\}$, $b_i \in \{0, 1\}$
The sending party outputs: $c = \{c_i = a_i \oplus b_i\}$
The ideal desiderata for this encryption algorithm are as follows:

1. An observer cannot distinguish c from a random string of bits
2. The receiving party can retrieve a from c

We can see that the first claim is true. Regardless of value of a_i, if b_i is chosen randomly, then c_i will look like a coin flip. This is easy to verify:

$$a_i = 1 \implies \begin{cases} c_i = 1 & b_i = 0, p = \frac{1}{2} \\ c_i = 0 & b_i = 1, p = \frac{1}{2} \end{cases}$$

$$a_i = 0 \implies \begin{cases} c_i = 0 & b_i = 0, p = \frac{1}{2} \\ c_i = 1 & b_i = 1, p = \frac{1}{2} \end{cases}$$

No matter what the bit of a_i is, c_i is 1 with probability $\frac{1}{2}$ and 0 with probability $\frac{1}{2}$. This means we fit the absolute ideal for encryption, that the attacker cannot distinguish c from a completely random string of bits.

Now we look to the second desideratum. An encryption scheme has no value if the receiver cannot retrieve the original message from our encrypted message. Thankfully, this is easy to do with one-time pass due to the associative nature of the exclusive or operation:

$$(a_i \oplus b_i) \oplus b_i = a_i \oplus (b_i \oplus b_i) = a_i \oplus 0 = a_i$$

Despite achieving both of our desiderata, this method has its drawbacks. We need to meet in a secret location every time we want to send a message, because the pads are randomly drawn for one-time use. If we were to re-use the same pad, and our opposition knew we were using a one-time pad strategy, information from the original message would become public. In addition, storing these pads doubles the size of every message we want to send, as the encryption string b must be as long as the original string a.

2.2 Rivest-Shamir-Adleman (RSA) System

The RSA system is another method of cryptography.[1] Two prime numbers p and q represent the secret key and $N = p \cdot q$. From p and q, we can compute exponents e (for encryption) and d (for decryption). We can then present our information x as $x^e \mod N$ – without using p and q specifically, so it looks like a random string to an outside observer. Someone who knows the value of d can compute $(x^e \mod N)^d \mod N = x$, which results in the original expression because of how e and d are calculated. This is an easy computation if you know d and e, but these values can only be determined if you know p and q.

In theory, one can obtain p and a simply by factoring N. However, all currently known factoring algorithms take exponential time, making them intractable problems to compute. So, RSA is protected by the speed (or rather slowness) of factoring a large number. This means that if someone were to find a factoring algorithm which worked in a reasonable time, RSA would be a much more vulnerable system. Unlike many algorithmic challenges, it is not true that if we find a faster factoring algorithm then $P = NP$. However, much work has been done on the subject, and it is not likely we can find a faster than exponential algorithm.

One final twist makes RSA especially appealing in comparison to the one-time pad. If p and q are randomly generated privately, N and e are published, and d is kept private, the recipient

[1]https://en.wikipedia.org/wiki/RSA_(cryptosystem)

can generate p and q by themselves, avoiding any private meeting. This is known as *public key cryptography*, which is often seen on the internet today. Some people have massive public keys on their websites, allowing them to receive secure messages. You only need N and e to encrypt a message, but you need d to decrypt it. Because of this, RSA is a one-directional form of encryption. To send a message, all you need to do is look up the recipients public N and e to encrypt your message, and then they'll use their private d, p, and q to decrypt it.

We talk about cryptography more generally to discuss the concept of proving things in mathematics, whether that be proving your identity or that something was not forged (using a private key to sign documents). We can also talk about the concept of a *zero-knowledge proof.* Let's say I want to prove to you that I have a valid credit card number. One way to do this is to simply tell you my Mastercard credit card number and have you verify it. But of course this solution has some issues, as you now have my credit card number. In cryptography, I want to convince you that I do in fact have a Mastercard **without giving you the number**. I want to prove to you that I know something without telling you what I know.[2]

3 Privacy

We now look at how we can keep databases private. This definition of privacy is more about keeping information hidden from what we publicly release, rather than the cryptographic notion of trying to send messages without an observer being able to interpret them. The desiderata for this problem can be described as follows:

1. We have some algorithm which will compute and publicly output a summary or variant of a database
2. Everyone can see this output, but we want to protect the privacy of the data in the database using some notion of anonymity
3. This anonymity to individuals in the database should protect them from being identified from the summary/variant/output database

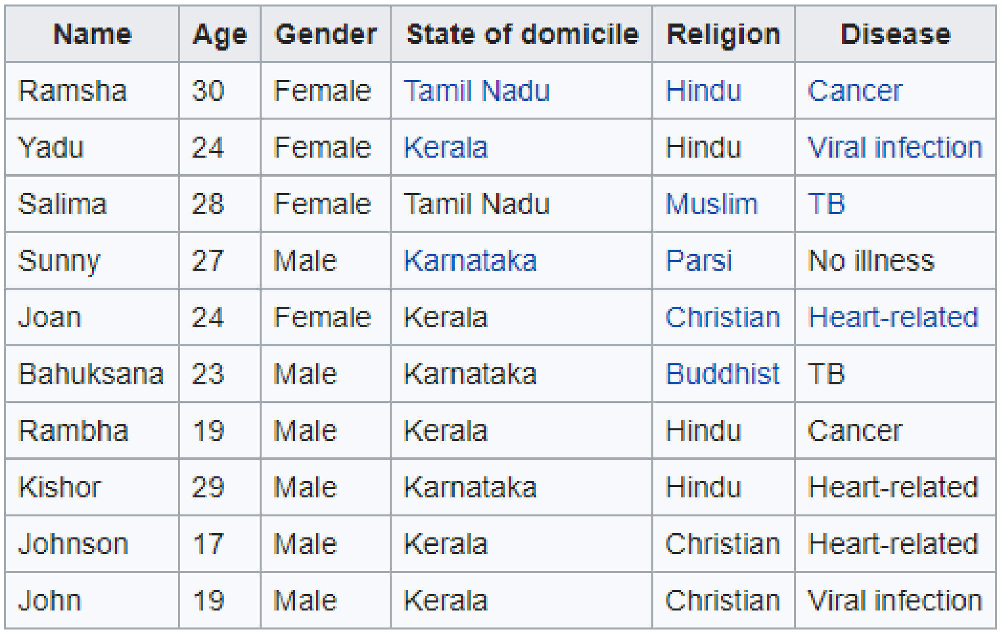
3.1 k-anonymity

Let's say we want to anonymize a database D. We want to compute and publicly release a *useful* version of D, call it D', that does not let you know about the specific contents of D. We can say that D' is **k-anonymous** if for every row in D', there are $\geq k$ identical rows. Imagine the various specifications: certain columns could contain "sensitive" attributes like smoking habits or medical test results. This database could be k-anonymous such that those protected columns have k copies. This is a flawed definition of privacy, but why is it bad? [3]

Consider the medical database taken from the Wikipedia page on k-anonymity. This database does not meet k-anonymity for any interesting value of k because there are no two people with the same name. Remember that k is a parameter such that as it increases, anonymity becomes more and more indistinguishable. You could remove all personal identifying information and the database could still only be 1-anonymous and contain a lot of information about you.

[2]https://en.wikipedia.org/wiki/Zero-knowledge_proof

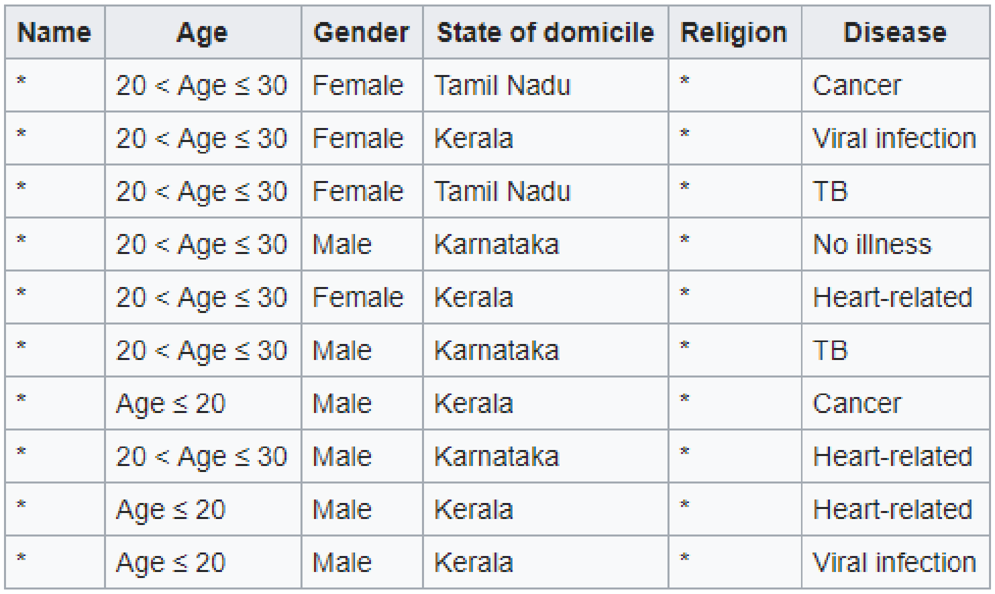
[3]https://en.wikipedia.org/wiki/K-anonymity



Name	Age	Gender	State of domicile	Religion	Disease
Ramsha	30	Female	Tamil Nadu	Hindu	Cancer
Yadu	24	Female	Kerala	Hindu	Viral infection
Salima	28	Female	Tamil Nadu	Muslim	TB
Sunny	27	Male	Karnataka	Parsi	No illness
Joan	24	Female	Kerala	Christian	Heart-related
Bahuksana	23	Male	Karnataka	Buddhist	TB
Rambha	19	Male	Kerala	Hindu	Cancer
Kishor	29	Male	Karnataka	Hindu	Heart-related
Johnson	17	Male	Kerala	Christian	Heart-related
John	19	Male	Kerala	Christian	Viral infection

An example database. This does not meet k-anonymity for $k \geq 2$, because at least one row is unique.

Let's propose a modification of the database where we remove the name and religion columns and "bin" or "coarsen" ages (meaning that two people aged 24 and 28 are now both aged "20-30"). We now have less information, but the total number of data points remains the same. Now, we have 2-anonymity with respect to age, state, and gender. This definition is precisely related to this database, as we have achieved 2-anonymity while keeping utility of our database. In general, it's very easy to meet a contrived definition where the output is just a new database with k identical rows and no additional restrictions.



Name	Age	Gender	State of domicile	Religion	Disease
*	20 < Age ≤ 30	Female	Tamil Nadu	*	Cancer
*	20 < Age ≤ 30	Female	Kerala	*	Viral infection
*	20 < Age ≤ 30	Female	Tamil Nadu	*	TB
*	20 < Age ≤ 30	Male	Karnataka	*	No illness
*	20 < Age ≤ 30	Female	Kerala	*	Heart-related
*	20 < Age ≤ 30	Male	Karnataka	*	TB
*	Age ≤ 20	Male	Kerala	*	Cancer
*	20 < Age ≤ 30	Male	Karnataka	*	Heart-related
*	Age ≤ 20	Male	Kerala	*	Heart-related
*	Age ≤ 20	Male	Kerala	*	Viral infection

After altering our previous database, we have achieved 2-anonymity with meaningful data still in the database.

'How can we ensure that k-anonymity keeps information? One operation on the database is deleting a feature entirely, while another is coarsening, where you replace a continuous quantity like age with different buckets. Of course, we *could* trivially delete every column so that every row is identical, but while that would meet our definition, the output would be useless. Therefore, we always want to do the *minimum* amount of coarsening and redacting necessary in order to get a k-anonymous database, ensuring that our output database is as similar to the original one as possible.

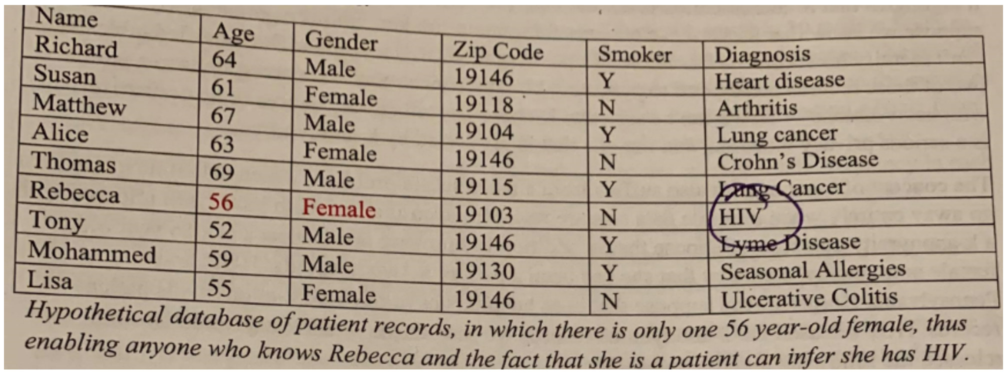
How would you write an algorithm that transforms a database D into a k-anonymous database D'? Perhaps an iterative algorithm that checks if D has a certain property (in this case k-anonymity), and if not, find a row that doesn't have k copies and do a sequence of operations on that. This practice is NP-complete, but there are approximation/computation algorithms which can be useful in practice. (See more on Wikipedia page linked above)

3.2 Problems with k-anonymity

What are the algorithmic problems with this definition? There's a utility problem: let's say you can find everything in reasonable time. What do you actually know about the original D from D'? You have no idea because there's no utility guarantee. Depending on what summary statistics you want, this strategy could be very bad.

Even ignoring utility concerns, there's a much bigger fundamental problem. Let's say Penn released medical records using 1000-anonymity. The conceptual problem of k-anonymity is that it assumes the only thing you're worried about is what will happen to you from this 1000-anonymous database, which isn't really the fundamental issue.

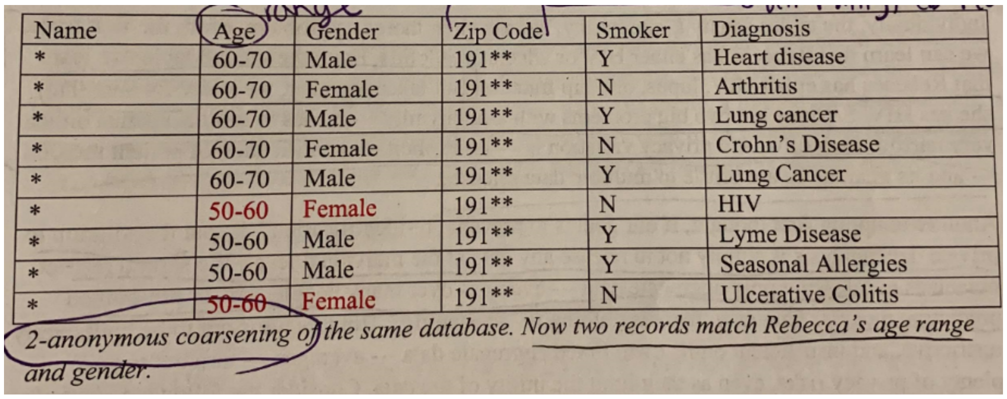
Let's make this concept clear using another example from Prof. Kearn's forthcoming book, "The Ethical Algorithm." Say there's a patient, Rebecca, who is your neighbor, so you know her age, gender, and ZIP code. If you looked in the database below of hospital patients, you'd clearly be able to determine Rebecca has HIV.



Name	Age	Gender	Zip Code	Smoker	Diagnosis
Richard	64	Male	19146	Y	Heart disease
Susan	61	Female	19118	N	Arthritis
Matthew	67	Male	19104	Y	Lung cancer
Alice	63	Female	19146	N	Crohn's Disease
Thomas	69	Male	19115	Y	~~Lung~~ Cancer
Rebecca	56	Female	19103	N	HIV
Tony	52	Male	19146	Y	~~Lyme~~ Disease
Mohammed	59	Male	19130	Y	Seasonal Allergies
Lisa	55	Female	19146	N	Ulcerative Colitis

Hypothetical database of patient records, in which there is only one 56 year-old female, thus enabling anyone who knows Rebecca and the fact that she is a patient can infer she has HIV.

Our first instinct here is to redact first names and partial ZIP codes (we leave them as "191xx" since this represents all of Philadelphia). This gives us a 2-anonymous database:



Name	Age	Gender	Zip Code	Smoker	Diagnosis
*	60-70	Male	191**	Y	Heart disease
*	60-70	Female	191**	N	Arthritis
*	60-70	Male	191**	Y	Lung cancer
*	60-70	Female	191**	N	Crohn's Disease
*	60-70	Male	191**	Y	Lung Cancer
*	50-60	Female	191**	N	HIV
*	50-60	Male	191**	Y	Lyme Disease
*	50-60	Male	191**	Y	Seasonal Allergies
*	50-60	Female	191**	N	Ulcerative Colitis

2-anonymous coarsening of the same database. Now two records match Rebecca's age range and gender.

Using this database, we can conclude that Rebecca has either HIV or Colitis — still a lot about her, but there's at least some ambiguity about a sensitive variable. But Rebecca doesn't even want you to know the realm of the ambiguity, and here's the real issue. Rebecca also visited a different hospital whose database that is 3-anonymous (the same as above, but ages are coarsened). The real issue comes here: you can join the two anonymous databases together and triangulate what Rebecca has, since the databases are not anonymous with respect to each other. So even if the database is very large or k is very big, there's just too much information available. If you hear of databases like these, be skeptical and concerned! Many breaches of privacy these days happen through this notion of re-identification that combines data from multiple sources. This is the fundamental problem with k-anonymity that stops it from being a particularly useful definition of privacy.

3.3 Second proposal for privacy

Can we delineate the "bad things" and/or "attacks" we are trying to prevent? In 1977, Tor Delaneus said: let D be any database, and anything that somebody can learn about you from D, should be something that they can learn about you without access to that dataset. So essentially, all the data was public to begin with. We can promise that for all of the worlds data we can figure out everything about you without having access to any of that data. This definition is *too strong* — it tries to protect us from so much that we can't actually do anything useful.

3.4 A Thought Experiment

Suppose it's 1950 and you're a smoker, along with nearly everyone else (there's no social stigma to smoking, just ask Kearns' parents). Would you be willing to let your medical records be used in a study which eventually causes the correlation between smoking and lung cancer? What if your insurer knows this correlation and the fact that you're a smoker, since you made no efforts to hide this? Your premiums would skyrocket because your chances of lung cancer are now significantly higher than before. Financial harm came to you as a result of a study, but this harm was going to come to you whether you contributed your data or not. Consider the database which has the original data, and one with the original data subtracted by your data. You withholding your data doesn't change the fact that smoking causes cancer, but a good definition of privacy protects you from harms that result in your record specifically being included in the database, but not from the results of the aggregate study as a whole. This the idea of **differential privacy**, which is explored in the following lectures.