Anonymization

27 October 2019 13:29

Pseudonymization:

- secret function to map **direct identifiers**
- can approximate polynomials
- use cryptographic hash instead
  - can be broken using lookup table
    - with guessing of hash function
    - length of original direct id
  - assume hash function known:
  - add salt to original direct identifier
  - **Salt:** fixed string of arbitrary length (but long!) that is added to the identifier before hashing it -must be kept a secret

Sensitive information: trying to protect

Identifier: directly identifies a person

Quasi-identifier:

- does not directly identify a person
- multiple taken together could uniquely identify a person

Auxiliary information: information known to an attacker

Uniqueness w.r.t A:

- fraction of the dataset that is uniquely identified by the set of A of quasi-identifiers

k-anonymous:

- every record in the table is indistinguishable from at least k-1 other records, with respect to every set of quasi identifiers

Equivalence class:

- set of records that have the same values for all the quasi-identifiers

Achieving k-anonymity:

- Non-perturbative methods
  - generalization - replace attribute values with more general ones
    - e.g. 43221 -> 4322* -> 43***
  - Suppression: delete a column or row
- Perturbative methods
  - add noise
  - data swapping
    - swap attributes between individuals

Homogeneity attacks:

- individuals in the same equivalence class all have the same sensitive attribute value
- prevention: $\ell - Diversity$
  - an equivalence class is ℓ-diverse if it contains at least ℓ distinct values for the sensitive attributes
  - a table is ℓ-diverse if every equivalence class is ℓ-diverse

Semantic attacks:

- sensitive attributes of individuals in an equivalence class are distinct but *semantically* similar

Skewness attacks:

- the distribution of sensitive attributes in a class in skewed

t-closeness:

- **distance** between the distribution of a sensitive attribute in the equivalence class and the distribution of this attribute in the whole table in nor more than a threshold **t**
- table has **t**-closeness if all classes have t-closeness

Big Data Anonymization

27 October 2019 13:30

- no sensitive attribute
- no quasi-identifier

<u>Unicity ϵ_p:</u>

- average fraction of the users in the dataset that are uniquely identified by **p** random points

Estimating unicity:

- Random set on N users
- for u in users
  - draw **p** points at random
  - is the trace unique?

<u>Reducing with generalization:</u>

- coarsen the data by reducing resolution
- helps, but is not sufficient

<u>Matching Attacks:</u>

- auxiliary info might not directly match data
  - noise, missing, multiple matches
- between two datasets
  - Anonymized dataset
  - Dataset with direct identifiers
- Rely on
  - measure of *distance* between two points
  - *linking algorithm*

Matching attack: location data

- Assumption - number of actions a user performs in a region at a time interval: $A(u,l,t) \sim Po(\lambda_{l,t})$
- Location - anonymized
- Actions - directly identifiable
- Step 1:
  - for each u_{DI} *and* v_{anon} compute a **score**
- Step 2:
  - Compute **max weight matching** between U and V users, using Hungarian Algorithm
- Step 3:
  - An edge for user U_{DI} is only considered a match if it is **significant** ($score > \epsilon * \sigma_{Udi\ edges}$)

<u>Profiling Attacks:</u>

- Identifying users in an anonymous dataset using an identified dataset collected at a different time
- Step 1:
  - Extract a profile of the user in the identified dataset
  - through a profiling distance/algorithm
    - Jensen-Shannon divergence
    - Dot product
    - Cosine similarity
    - L_1
- Step 2:
  - Compare the profiles of known users to users in the anonymous dataset
  - to identify them using a linking algorithm
    1. Compute histograms in both datasets
    2. Compute the distance between each pair of histograms
    3. Use Hungarian algorithm to fin the max weight matching
       1) same condition for 'good' match

Query-Based Systems

07 November 2019 12:53

<u>QBS:</u>

- Don't share the database.
- Provide aggregates for statistical purposes
  - e.g. counting queries
- Just QBS
  - Susceptible to uniqueness attacks

<u>Query Size Reduction:</u>

- Query set must be over a certain threshold
- If not, don't return value
- Susceptible to **intersection** attacks

<u>Bounded Noise Addition:</u>

- Perturb output of every query
- If adding non biased noise
  - $\tilde{A} = A + [-N, +N]$
  - $E(\tilde{A}) = A$
- Attacker Knows the bound
  - Calc diff between two queries
  - Make use of unique Aux info
  - Figure out A and A
- Groups
  - If Group share a secret attribute
  - Can find out the value of the secret

<u>Unbounded:</u>

- Solves above two issues
- Centered at 0 - don't introduce bias
- Bayes' Theorem
  - $P[A = x | \tilde{A} = O] = \frac{P[\tilde{A} = 0 | A = x] * P[A = x]}{P[\tilde{A} = O]}$
- Averaging Attacks
  - CLT
  - Bayes' - also gives posterior distribution
  - **Required number** of queries:
    - $\boldsymbol{n \geq 4\sigma^2 z_\alpha^2}$
    - $\boldsymbol{z_{0.05} = 1.96}$, $\boldsymbol{z_{0.01} = 2.58}$
  - Defend
    - Add consistent noise
    - Cache query - return cached value
    - Seeded PRNG
  - Get around defense
    - **Semantic averaging attacks**
    - Logically Equivalent Query but expressed differently

Diffix:

- Sticky noise
  - For each condition
  - Add static
    - Random value
    - Seeded
      - $hash(C, salt)$
  - Add dynamic
    - Seed hash
      - Static seed
      - Unique ID of all conditions in the Query
- Bucket Suppression (QSR)
  - Dynamic, noisy threshold
  - If $Q(D) \leq 1 \rightarrow suppress$
  - Else $T \sim N(4, 0.5)$
    - Seeded hash
    - Salt and UID of C in Q
    - Suppress if $Q(D) < T$
- Split Averaging Attacks
  - Pair of Semantically same query
  - Across Range of an attribute
  - Average these to get desired
- Diffix blocks by having static on each condition

Differential Privacy

12 November 2019 13:28

$$1 - \epsilon \preccurlyeq \frac{\Pr(output = y \mid x \in D)}{\Pr(output = y \mid x \notin D)} \preccurlyeq 1 + \epsilon$$

Neighboring Dataset:
- D1 neighbors D2 if they differ by only one row

Formal DP:
- $\Pr(M(D) = y) \le \mathrm{e}^{\epsilon} \Pr(M(D') = y)$
  - $e^{\pm\epsilon} \approx 1 \pm \epsilon$

Hard to protect **against group attacks** with DP
- Scale with group size **($\boldsymbol{k\epsilon}$)**

Achieving DP:
- $f_X(x|\mu, b) = \frac{1}{2b} exp\left(-\frac{|x-\mu|}{b}\right)$
  - $\sigma^2 = 2b^2$
  - Like a sharper Gaussian
- $Lap(b) = f_X(x|0, b)$
- For one query
  - **Add $\boldsymbol{Lap(1/\epsilon)}$**

Protect against Averaging attacks - **Composability**:
- Releasing output of any two queries protected by $\epsilon - DP$
- Same as releasing one query protected by $\epsilon - DP$

Privacy Budget
- Add $Lap(1/\epsilon_i)$ to Q_i
- With the constraint: $\epsilon = \sum_i^n \epsilon_i$

Optimizing DP: **Histograms** (counts):
- Add $Lap(1/\epsilon)$ to every bucket, as opposed to multiple queries
  - Since a user only contributes to one bucket
  - Don't need to split budget

Optimizing DP **Function Sensitivity**:
- Global sensitivity of a function f
  - $\Delta f = \max|f(D) - f(D')|$ (for any D and D')
- Add noise $Lap\left(\frac{\Delta f}{\epsilon}\right)$

Local DP:
- User gives **randomizes response**
- $\Pr(M(r_1) = r) \le \mathrm{e}^{\epsilon}\Pr(M(r_2) = r)$

Computing on Encrypted Data

01 December 2019 20:49

Secure Multi-Party Computation:

- Private inputs
- Jointly compute over a function
- One or more decrypt output

Multiparty Addition Protocol

- Split shares
- Send to parties not named in share
- Broadcast **partial sums**

Yao's Millionaire problem:

$Z[i] = Priv_A(Pub_A(r_{bsecret} - b) + i) \ mod \ p_{rand}$

- $z[i] += 1 \ for \ i > a$
- Z values all differ by at least 2 from each other
  - So that post increment they differ by at least 1

Honest but curious

- Follows protocol but opportunistic (can collude)

Malicious

- Deviates from protocol (can collude)

Asynchronous Communication

- Results sent asynchronously

Fair

- Secure against not forwarding last message

Oblivious Transfer Protocol (1 from 2):

- A->B Two public keys
- B choses one, and send symmetric K
- A decrypts twice, sends $K(m1), K_{bad}(m2)$
  - Other way round if B chose 2nd pubA
- Bob decrypts, with K, same choice

Yao Garbled Gates (B doesn't know gate function)

- A create two random keys for each wire
- A computes truth table, sends **shuffled** one to B
- A sends choice of A keys to B, (A inputs)
- Using an **OT, B selects a KBX for each wire**
- A tells B final K mappings

- $\boldsymbol{k[w,0]p[w]}$ **encrypts a 0**
- $\boldsymbol{k[w,1] not \ p[w]}$ **encypts a 1**

- Permute bits index the table
- Perform gate on encoded values
- Selects new permute

- B just index's and is told what keys to use to decrypt

One Time Memory:

- Store secret K0 and K1
- Select with input
- Third bit, x, says if used

One Time Program:

- Convert to Yao Garbled Circuit
- Use OTM to store k[]'s

Extending Yao to two circuits:

- Want to compute $f_B(a,b)$ as well as $f_A(a,b)$
- B computes
- $\boldsymbol{f(a,b,k)} = k \oplus f_A(a,b) \ \ f_B(a,b)$
- B sends first part to A
- A decrypts by XOR'ing with secret bit k

Computing on Untrusted Servers

03 December 2019 15:43

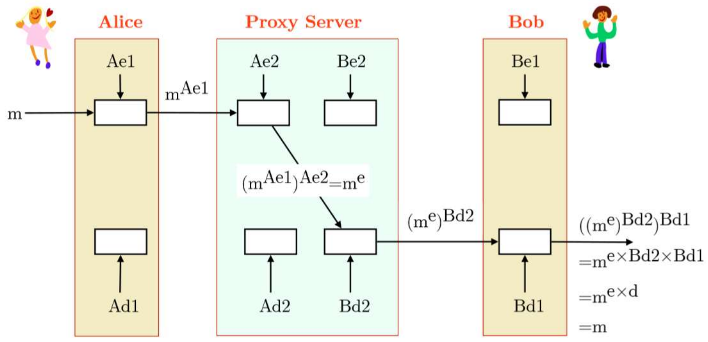
CryptDB:

- encrypt query
- Query made on encrypted data
- results decrypted

- **Onions** of Encryptions
  - multiple layers of encryption
  - CryptDB has different onions for different operators
  - an **attribute might have multiple** onioned values depending on application

- based on secret key shared by clients
  - (decrypt results)
- hard to remove a user

Proxy-based encryption:

- Key server generates a master RSA key par
  - $(e, n), (d, n)$
- For each user
  - a pair for user $(e1, n), (d1, n)$
  - a pair sent to DB server $(e2, n), (d2, n)$
  - $e1 * e2 = e \bmod (p-1)(q-1)$
  - $d1 * d2 = d \bmod (p-1)(q-1)$
- 

Symmetric Bilinear Pairings:

- pair two elements from one group to an element of a second group
- G_1, G_2 are two cyclic groups of prime order q
- g is a generator for G_1
- bilinear pairing $G_1 * G_1 \rightarrow G_2$
  - for all $u, v \in G_1, \quad a, b \in \mathcal{Z}_q$
  - $e(u^a, v^b) = e(u, v)^{ab}$
  - $e(g, g) \neq 1$
  - $e(u, v)$ computable

Anonymous Communication I

03 December 2019 15:44

<u>Chaum Mixes (Mix-Nets):</u>

- Mixer T - trusted
  - Forwards to A
  - $\boldsymbol{K_t(a, K_a(m))}$
  - reorders outbound forward
    - adds delay
- Listener knows number of inward and outbound, but can't match them

- Improve by doing $\boldsymbol{A \rightarrow t1 \rightarrow t_2 \rightarrow B}$
  - $K_{t1}\left(t_2, K_{t2}\left(b, K_b(m)\right)\right)$

- Anonymous Reply
  - $K_t\left(b, K_b\left(m, K_t(a, K_x), K_y\right)\right)$
  - K_x, K_y one time pair, or symmetric

<u>Onion Routing:</u>

- relay nodes
- e.g. $\boldsymbol{K_{r1}(r2, K_{r2}(\ldots(B, m)))}$

<u>Tor:</u>

- clients creates virtual circuits with **relay nodes**
- Inter relay encrypted using TLS
- 512 byte cell size

<u>Controls Cells</u>

- CID(2): Circuit ID for the link
- CMD(1):
  - CreateCreated/Destroy circuit
- PAYLOAD(509):
  - additional control data

<u>Relay Cells:</u>

- CID(2)
- CMD(1) = Relay
- Next encrypted with AES session key for this link (made during control)
  - MIsc(10)
  - CMD(1)
    - extend/extended
      - sends new AES keys for next relay back to A
    - begin/connected (TCP connection)
    - Data
  - Payload(498)

<u>Circuit Construction:</u>

- P is tor proxy for A
- P -> R1 Create
- R1-> P Created (AES K1)

- P -> R1 Relay, Extend R2
- R1 -> R2 Create
- R2 -> R1 Created (AES K2)
- R1 -> P Relay, Extended (K2)

<u>TCP</u>

- P -> R1 Relay (R2, Begin B)
- R1 -> R2 Relay (Begin B)
- R2 -> R1 TCP handshake B
- R2 -> R1 Relay Connected
- R1 -> P Relay Connected

<u>Location-Hidden Services:</u>

- B creates Tor circuit to **Introduction Relays**
  - Inform of **Long-term Public** key
- Publish Name, Public Key, and IR's to **lookup service**
- A finds this then
  - Asks via **IR** for B to connect to A via
    - *a rendezvous relay* **V** (separate from IRs)

Anonymous Communication II

03 December 2019 23:57

Attacks against Tor:

- **Global traffic analysis**
  - fingerprint B/W over time

- **Active interference attacks**
  - congest victim to relay
  - monitor which relay to public server flow is affected
  - Overcome - **Collective Control Plane** (CCP)
    - DC-nets for secret inputs to public outputs
    - managed by **CCP Policy Oracle**
      - control when/ how much to send

- **Denial-of-service**
  - on lookup service

- **Intersection attacks**
  - Intersects with users in non Tor that match quasi-identifiers
  - That also used Tor at a certain time
  - **Buddies**
    - Policy Oracle reports metrics on simulated intersection attacks
    - Possinymity: possibilistic deniability
      - intersection of users for a Nym for a time period
    - Indinymity: probabilistic indistinguishabiliity

DC-Nets:

- Dining Cryptographers
- **XOR** on left, right, + your secret
- Send to middle - another **XOR**

- Doesn't scale well
- If 1 user leaves, have to restart
- 1 Malicious user can jam communication

- **De-anonymizing exploits**
  - attack the browser

- **Accountability provisions**
  - Before Unmasking as last resort
    - threat of censure
    - give an opportunity to retract
    - expulsion from group

Privacy Policies

04 December 2019 10:56

S4P Language:

- assumes user trust service providers to enforce user policies
- supports policy evolution

User:

- may assertion -> gives permission
- will query -> asks for promise

Service:

- may query -> asks for permission
- will assertion -> gives promise

For satisfactions**, compare** mays and wills (**queries to assertions**)

Both a **user privacy preference** and **service privacy policy** consist of a set of assertions and a query.

Assertion

```
E says f0 if f1, .., fn where c
```

Delegation of authority

Fact

```
f ::= a | e may b | e will b | e can say f
```

Query

```
q ::= e says f? | c? | ¬q | q1 ∧ q2 | q1 ∨ q2 | exists x: q
```

E	constant	Typically **principals** e.g. Alice, Bob, Service Provider
x	variable	
e	expression	**Constant** or **variable**
c	constraint	Constraint on variables occurring in assertion
a	atom	**Predicate** written in *infix* notation e.g. 'Alice is a nicePerson'
b	behaviour atom	**Service behaviours** e.g. 'delete email within 1 yr'