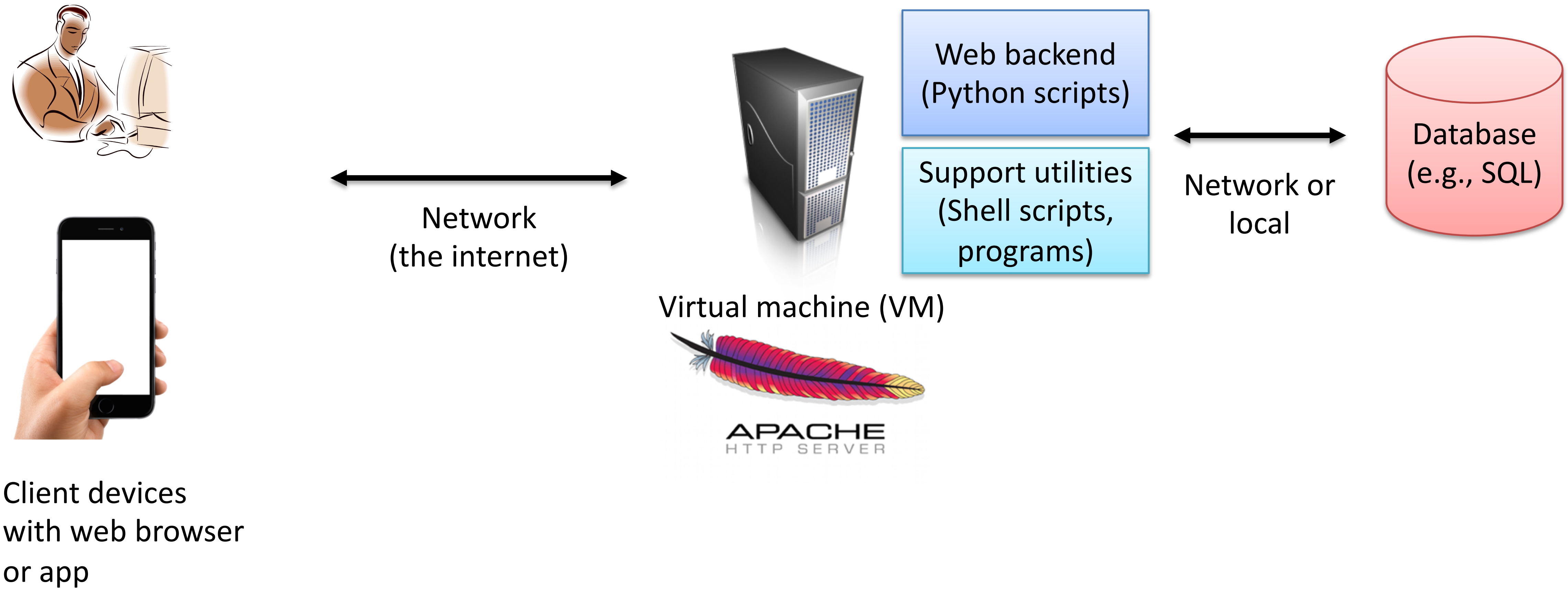
# CS 5435: Data protections

Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2019

# Attacks that can expose sensitive data?



# Today: how do we minimize damage of breach?

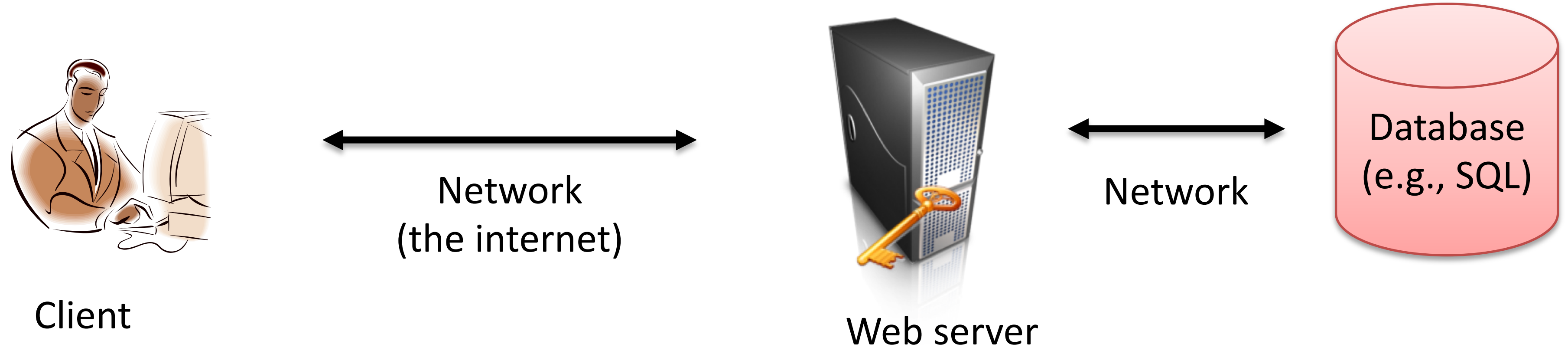
- Access controls
  - Least privilege principle applied to database
  - Monitoring & logging accesses
- Encrypted databases
  - Encryption at rest
  - Property-revealing encryption
- Data privacy protections
  - De-identification
  - K-anonymity
  - Differential privacy

# Encrypted databases

- Encrypting data-at-rest
  - Encrypt before storing to persistent storage (keys in software)
  - Encrypted hard drives (keys stored in hard drive)

# Encrypted databases

- Encrypting data-at-rest
  - Encrypt before storing to persistent storage (keys in software)
  - Encrypted hard drives  (keys stored in hard drive)
- Encrypting *before* insertion into DB
- What attacks could these prevent?



# Example: outsourced storage settings


Salesforce stores customer records for companies

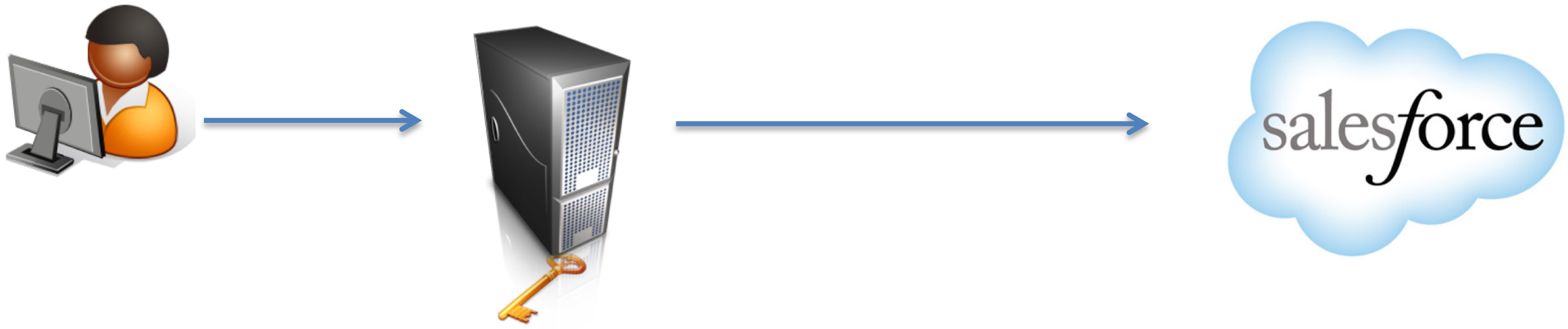
| | | | |
|---|---|---|---|
| Name: | Clarisse | Comments: | Works in NYC office |
| Age: | 22 | Gender: | Female |
| Salary: | 100,000 | SSN: | 555-31-4325 |

Much value-add server-side functionality

- Keyword search (find all records with name “Clarisse”)
- Range queries (all records with 20 <= Age <= 30)
- Sorted lists (return records orderered by salary)

What security threats would one worry about?

# Example: outsourced storage settings




Salesforce stores customer records for companies

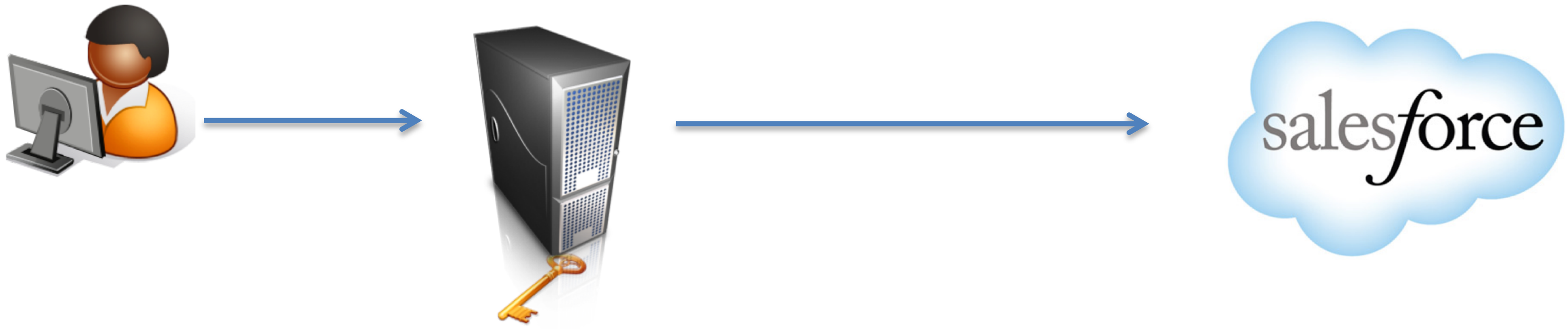
Name:
Age:
Salary:

Comments:
Gender:
SSN:

What DB functionalities broken if we use standard authenticated encryption?

- Field search (Find all records with Name = Alice)
- Keyword search
- Range queries (Find all people who make between 90k and 120k)
- Format problems (Age must be integer between 0 and 130)
- …

# Example: outsourced storage settings




Salesforce stores customer records for companies

Name:
Age:
Salary:

Comments:
Gender:
SSN:

CipherCloud™ Trust in the Cloud

PerspecSys Making the Public Cloud Private

skyhigh

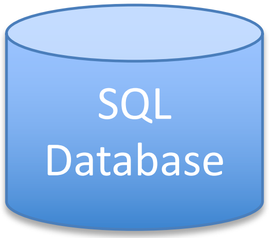
One approach:
Encrypt data with special ***property-revealing encryption (PRE)*** schemes that leak just enough about plaintexts to perform some operations

Cloud access security brokers (CASB)

# Property-revealing encryption (PRE)

| Problem | Crypto primitive | Description | Literature |
|---|---|---|---|
| Keyword search | Searchable symmeric encryption | Perform search over ciphertexts given encrypted search token | [Dawn, Song, Wagner 2000] [Curtmola et al. 2006] … |
| Equality search | Deterministic encryption | X = Y implies DET(X) = DET(Y) | [Rogaway Shrimpton 06] |
| Range queries | Order-preserving encryption | X > Y implies OPE(X) > OPE(Y) | [Boldyreva et al. 2009] |
| Range queries | Order-revealing encryption | X > Y implies Cmp(ORE(X),ORE(Y)) = 1 | [Boldyreva et al. 11], [Boneh et al. 15] |
| Deduplication | Message-locked encryption (Convergent encryption) | Different user’s encryptions of same plaintext give same ciphertext | [Douceur et al. 2002] [Bellare et al. 2013] |
| Format restrictions | Format-preserving encryption | Ciphertext has same format as plaintext | [Bellare et al. 2009] |

# PREs in databases

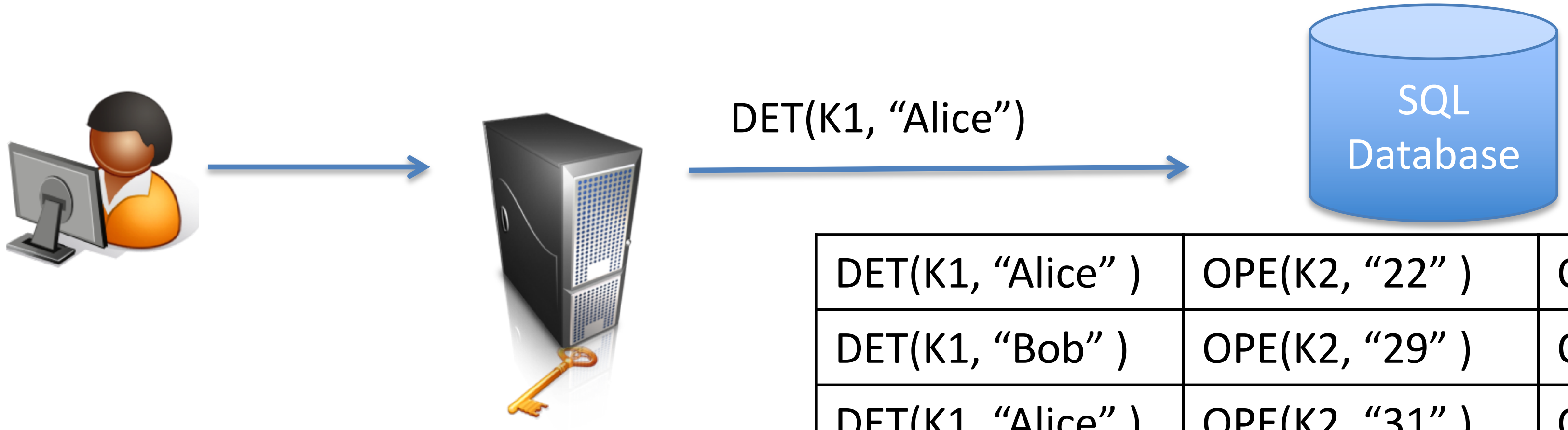



| | | |
|---|---|---|
| DET(K1, “Alice” ) | OPE(K2, “22” ) | OKE(K3,“100,000”) |
| DET(K1, “Bob” ) | OPE(K2, “29” ) | OPE(K3,“80,000”) |
| DET(K1, “Alice” ) | OPE(K2, “31” ) | OPE(K3,“130,000”) |

Encrypt by column

Name: DET(K1, “Alice” )
Age: OPE(K2, “22” )
Salary: OPE(K3, “100,000” )

# PREs in databases



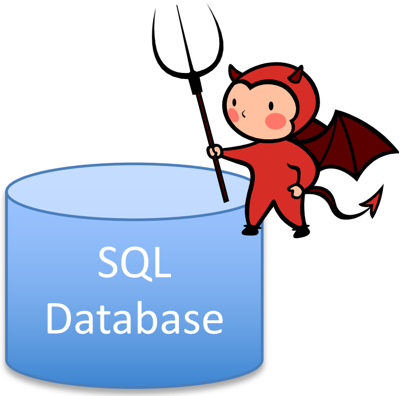

| | | |
|---|---|---|
| DET(K1, “Alice” ) | OPE(K2, “22” ) | OKE(K3,“100,000”) |
| DET(K1, “Bob” ) | OPE(K2, “29” ) | OPE(K3,“80,000”) |
| DET(K1, “Alice” ) | OPE(K2, “31” ) | OPE(K3,“130,000”) |

Perform *equality search* by querying DET encryption of keyword

$$\mathrm{DET}(K1,X) = \mathrm{DET}(K1,Y) \;\Leftrightarrow\; X = Y$$

# PREs in databases

OPE(K2, 10) , OPE(K2, 30)




| | | |
|---|---|---|
| DET(K1, “Alice” ) | OPE(K2, “22” ) | OKE(K3,“100,000”) |
| DET(K1, “Bob” ) | OPE(K2, “29” ) | OPE(K3,“80,000”) |
| DET(K1, “Alice” ) | OPE(K2, “31” ) | OPE(K3,“130,000”) |

Perform *equality search* by querying DET encryption of keyword

$$\mathrm{DET}(K1,X) = \mathrm{DET}(K1,Y) \iff X = Y$$

Perform *range search* by querying OPE encryption of end points

$$\mathrm{OPE}(K2,X) < \mathrm{OPE}(K2,Y) \iff X < Y$$

What is revealed to adversarial server?

# Frequency analysis

| |
|---|
| DET(K1, “Alice” ) |
| DET(K1, “Bob” ) |
| DET(K1, “Alice” ) |
| DET(K1, “Alice” ) |
| DET(K1, “Alice” ) |
| DET(K1, “Bob” ) |

Auxiliary data is plaintext dataset distributed similarly to target. Histogram:

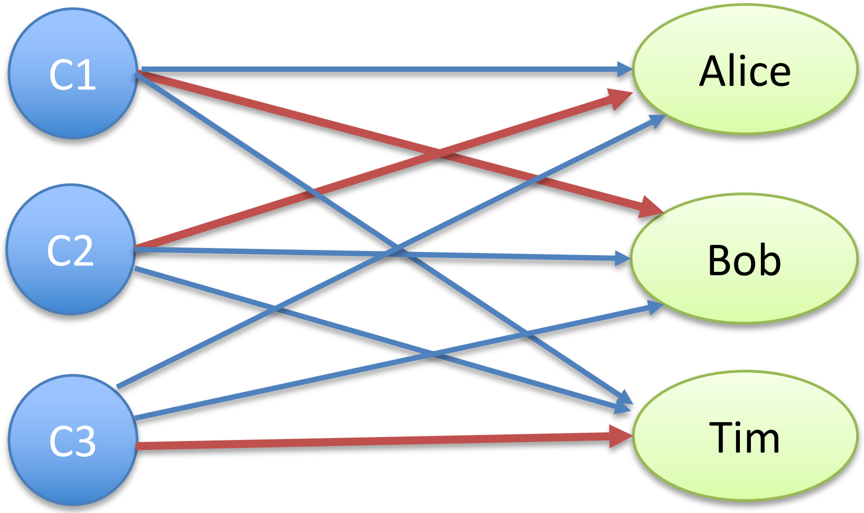
Alice 15
Bob 3
…

p(Name) = Freq(Name)/N
p(Alice) = 15/100000, p(Bob) = 3/10000, …

Match most frequent ciphertext to most frequent auxiliary value
Match next most frequent to next most frequent, etc.

Technique is thousands of years old; applied most often to substitution ciphers
But: DET is just a substitution cipher over large alphabet

# Graph viewpoint for attacking DET

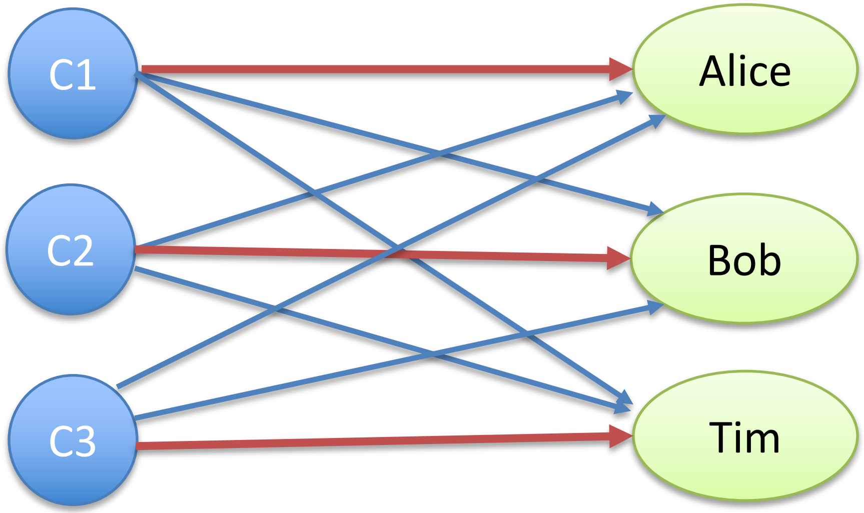



***Bipartite graph problem***: find matching of ciphertexts, names that maximizes total weight

$$\text{Weight}(C,\text{Name}) = \text{Freq}(C) * \log p(\text{Name})$$

Equivalent to frequency analysis

# Graph viewpoint for attacking ORE




***Bipartate graph problem***: find *non-crossing* matching of ciphertexts, names that minimizes total weight

$$\text{Weight}(C,\text{Name}) = \text{Freq}(C) * \log p(\text{Name})$$

This takes into account ordering constraints

Dynamic programming algorithm efficiently finds non-crossing matching

# Leakage-abuse attacks (LAAs)

Question: what can attackers learn from leakage of PRE schemes?

Searchable encryption

[Islam, Kuzu, and Kantarcioglu 2012]
[Cash, Grubbs, Perry, R. 2015]
[Zhang, Katz, Papamanthou 2016]
[Wright, Kamara 2016]
…

DET, OPE, ORE

[Naveed, Kamara, Wright 2015]
[Durak, DuBuisson, Cash 2016]
[Grubbs et al. 2016]
…

[Grubbs et al. 2016] recovery rates (percent of DB recovered) for ORE schemes:

| Scheme(s) | First names | Last names |
|---|---|---|
| Kerschbaum [27] | 26% | 6% |
| Popa et al. [36], Kerschbaum [28] | **84%** | 38% |
| BCLO [12, 13] | **99%** | **97%** |
| CLWW [18] | **98%** | **75%** |
| BCLO + CLWW [18] | **85%** | 44% |
| Baseline Guessing | 4% | 1% |

# Today: how do we minimize damage of breach?

- Access controls
  - Least privilege principle applied to database
  - Monitoring & logging accesses
- Encrypted databases
  - Encryption at rest
  - Property-revealing encryption
- Data privacy protections
  - De-identification
  - K-anonymity
  - Differential privacy

Can we limit the damage of leaking data by lowering its sensitivity?

# De-identification

https://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.8053.pdf

Remove identifiers from a database

| Name | Age | Salary |
|---|---|---|
| Alice | 22 | 100,000 |
| Bob | 29 | 80,000 |
| Charlie | 31 | 130,000 |

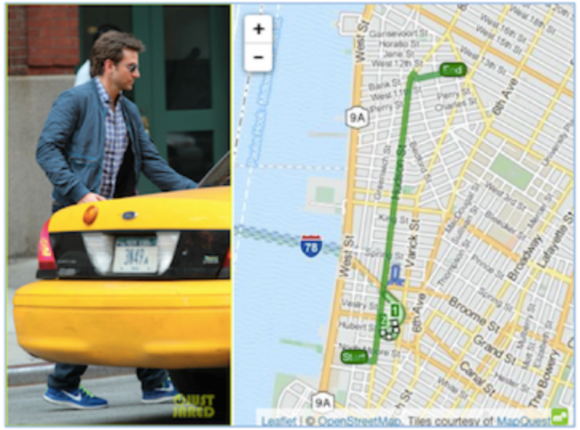
| Pseudonym | Age | Salary |
|---|---|---|
| 7412 | 22 | 100,000 |
| 8192 | 29 | 80,000 |
| 3841 | 31 | 130,000 |

How should pseudonym be generated?

NYC Taxi Commission released dataset of 173 million rides in 2014

Pseudonym = MD5(taxi ID)

Trivial to brute-force recover tax IDs



https://research.neustar.biz/author/atockar/

Better approaches:

- Randomly choose pseudonyms
- DET encryption, keyed hash (PRF), and throw away key

# De-identification & linkage attacks

Remove identifiers from a database rarely sufficient

| Name | Age | Salary |
|---|---|---|
| Alice | 22 | 100,000 |
| Bob | 29 | 80,000 |
| Charlie | 31 | 130,000 |

| Pseudonym | Age | Salary |
|---|---|---|
| 7412 | 22 | 100,000 |
| 8192 | 29 | 80,000 |
| 3841 | 31 | 130,000 |

What do I need to know to figure out which row corresponds to Alice?

Linkage attacks:

- Determine ***quasi-identifier*** for individuals
- Use pseudoidentifier to look up individuals in another database

[Sweeney 2000]: 87% of Americans uniquely identified by **ZIP code, sex, date of birth**

# De-identification & linkage attacks

Remove identifiers from a database rarely sufficient

| Name | Age | Salary |
|---|---|---|
| Alice | 22 | 100,000 |
| Bob | 29 | 80,000 |
| Charlie | 31 | 130,000 |

| Pseudonym | Age | Salary |
|---|---|---|
| 7412 | 22 | 100,000 |
| 8192 | 29 | 80,000 |
| 3841 | 31 | 130,000 |

What do I need to know to figure out which row corresponds to Alice?

[Narayanan, Shmatikov 2008]:

- re-identification attack robust to imprecise background info, perturbations in data
- Case study: re-identify people in de-identified Netflix prize dataset

# Countermeasures?

- Try to remove quasi-identifiers
  - Dataset is ***k-anonymous*** if for every combination of quasi-identifiers there are at least k matching records [Samarati, Sweeney 1998]
  - Can: suppress attributes, generalize attributes, add dummy records
  - Various generalizations such as L-diversity
- Generally considered not to provide strong privacy protections
  - Hard to know a priori what can serve as quasi-identifiers
  - Re-identification, other privacy attacks shown to still be possible

# Differential privacy

[Dwork, McSherry, Nissim, Smith '06]

Build randomized mechanisms for rendering database (or queries against a database) s.t. adversary can't with high confidence know if any particular individual in data set

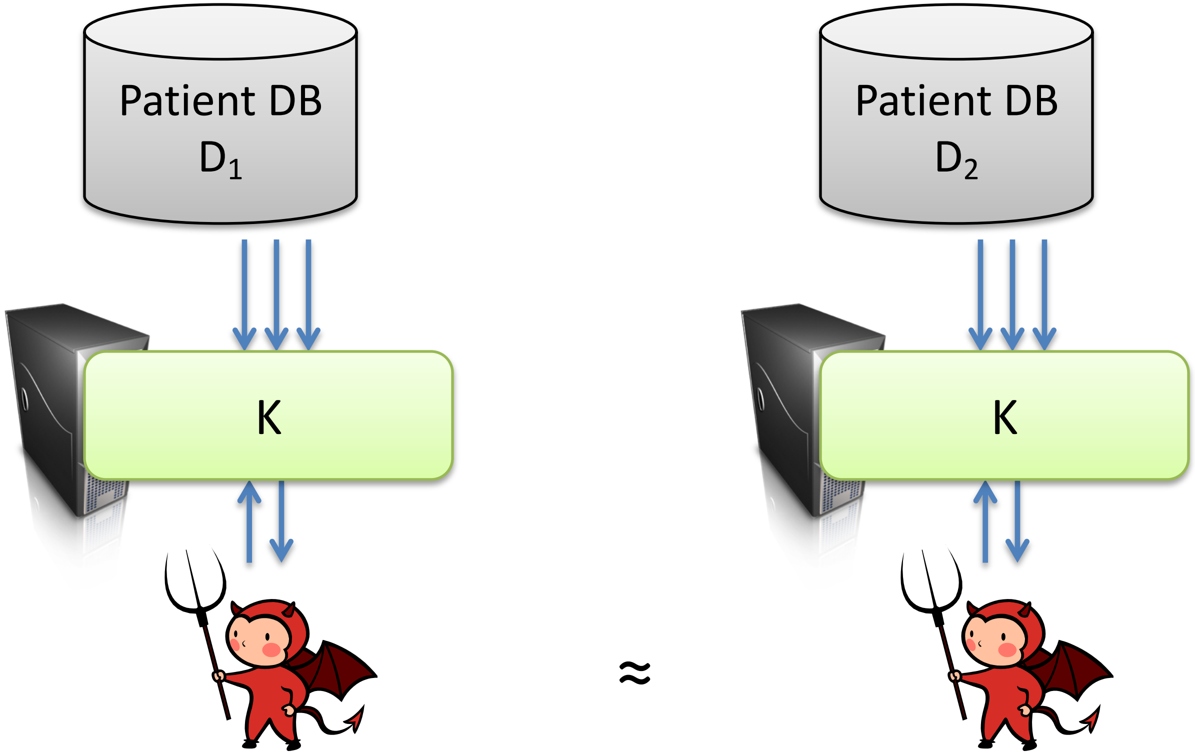
K is randomized algorithm. K is ***ε-differentially private*** if:
for all datasets $D_1$ , $D_2$ differing in at most one row and all subsets $S$ of K's range:

$$\Pr[\mathsf{K}(D_1) \in S] \le e^{\epsilon} \cdot \Pr[\mathsf{K}(D_2) \in S]$$

K *necessarily* adds noise, forcing a utility vs. privacy trade-off

# Differential privacy

[Dwork, McSherry, Nissim, Smith ‘06]



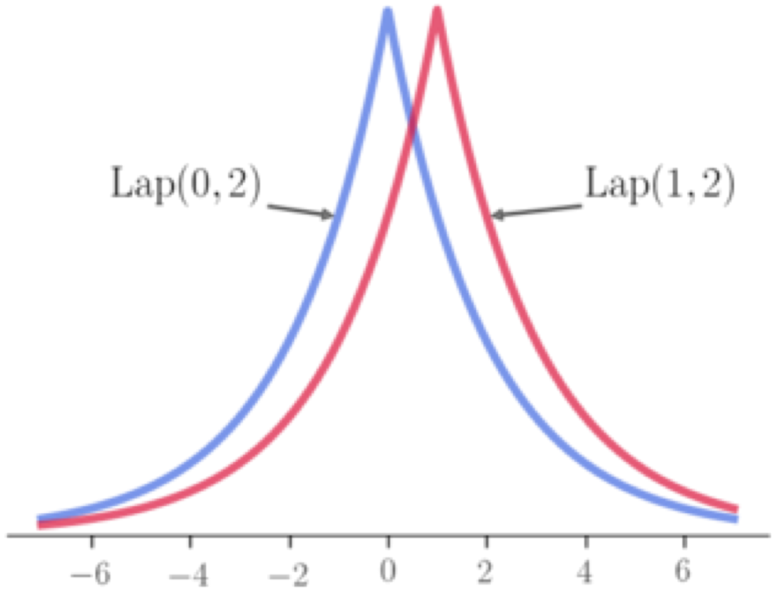
# Differential privacy

[Dwork, McSherry, Nissim, Smith '06]

Say we only want to render private some ***statistic*** of DB
Let f be function computing the statistic (e.g., median)

$$K(D,f,\varepsilon) = f(D) + \mathrm{Lap}(0,\Delta f/\varepsilon)$$

where $\Delta f = \max|f(D_1) - f(D_2)|$ is sensitivity f and
$\mathrm{Lap}(0,\Delta f/\varepsilon)$ is random according to Laplacian distribution




***Intuition:*** add sufficient random noise centered on actual value to ensure uncertainty about which database used

# Rappor system in Chrome browser

Google wants to collect information on websites visited by users of Chrome

- [Elringsson et al. 2014] gives DP system for sending reports
- Builds on ***randomized response***. Flip a coin:
  - Heads: report “Visited example.com”  no matter what
  - Tails: report whether this user visited example.com
- Uses variant with deniability for both Yes/No answers, long-term private response, short-term private response, …
- Rappor uses ε = ln 3

- Various limitations, see paper

# DP at Apple

Apple deployed DP mechanism for certain user data items collected by iOS

- Proprietary implementation
- [Tang et al. 2017] reverse engineered implementation:
  - Privacy budget epsilon ε not sufficient
  - Re-upped privacy budget each day
- Unclear how much privacy protection really offered

# Data protections

- Need multiple layers of defenses
  - ***Minimize sensitive data collected and stored***
  - Deny access to data (access controls + good software security)
  - Encrypt it as much as possible and use best practices for key management
  - Anonymization privacy techniques can be useful, but delicate