Module 5

Data Security and Privacy

Data Security and Privacy

- Data Security
  - Who has access to the data?
  - Who can change the data?
  - What are the threats to the data?
  - How do we mitigate the threats?
- Data Privacy
  - Who is the data about?
  - How can we share data without threatening people's privacy?

Access Control

- Define access control for (legitimate) users
- Mandatory vs Discretionary models 自由的
  - Mandatory: Policy controls all r/w/x permissions
    - Includes: Multi-level security
  - Discretionary: Each user decides
    - Includes: Unix file access control

Bell-LaPadula Model

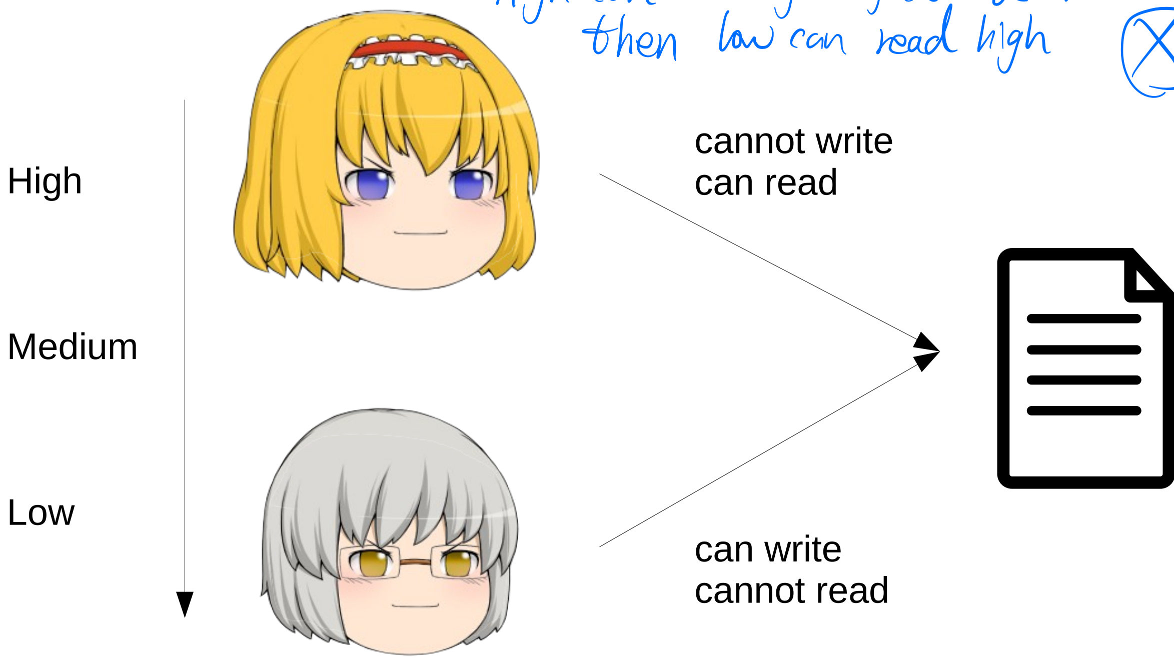
- Example of Multi-Level Security
- Subjects and Objects both have security levels (e.g. High, Low)
- All read/write must follow two rules (next slide)
- Prevents leakage of information (i.e. confidentiality)

Bell-LaPadula Model

1) A lower security subject cannot read a higher security object

2) A higher security subject cannot write to a lower security object

high can read high object and write to low
then low can read high X not allow!!!



Biba Integrity Model

- Like Bell-LaPadula, but reversed. Two rules:
  1) A higher security subject cannot read from a lower security object
  2) A lower security subject cannot write to a higher security object
- Prevents flow of incorrect information (i.e. integrity)

High-water and Low-water mark

- Replaces rule ~~2~~ 1) of each model
- High-water Bell-LaPadula: After higher security subject writes to lower security object, increase security level of object to level of subject
- Low-water Biba Integrity: After high security subject reads from lower security object, decrease security level of subject to level of object

File Access Control

- Access control matrix
- Access control list
- Capabilities
- Role-based

		Objects		
		Data1	Data2	Data3
Subjects	Alice	rw	r	-
	Bob	-	-	r
	Carol	r	r	r

Access Control List

- "Which subjects can read/write/execute this object?"
- e.g. chmod 744 on Unix (what does it mean?)

use file group rights

user's rights

else rights

first write binary

7: 111 → can read
can write
can execute

4: 100 → can read
x
x

4: 100 → can read
x
x

		Objects		
		Data1	Data2	Data3
Subjects	Alice	rw	r	-
	Bob	-	-	r
	Carol	r	r	r

ACL for Data1

Capabilities

- A transferable “reference” that gives a subject permissions to an object
- “Which objects can this subject read/write/execute?”
- f = open("filename", r);

		Objects		
		Data1	Data2	Data3
Subjects	Alice	rw	r	-
	Bob	-	-	r
	Carol	r	r	r

Alice's capabilities

Error detection

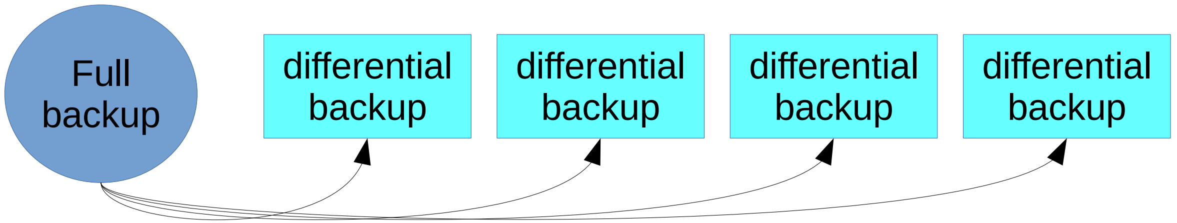
- Small number of bit errors should be detectable
- Append a tag to a file:
  - Parity → Add all bits % 2
  - Checksums, e.g. CRC32
  - Hashes; a weak cryptographic hash may be a good error detection hash (e.g. MD5)
- Input can be any size, output is fixed (32-bit for CRC32, 128-bit for MD5)
- Cannot fix an error

Backup

- Used for disaster recovery – we want to recover our data after corruption
- Full backups store all data, but we cannot store too many
- We need to use differential and incremental backups

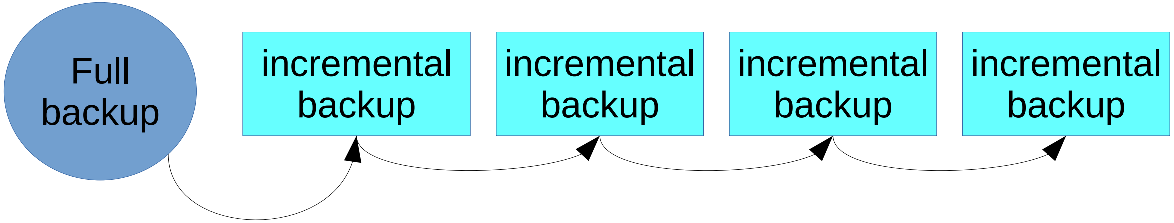
Differential backup

- Stores all changes between current time and last full backup
- How can we find changes?
  - e.g. rsync in Unix: Divide file into chunks, then hash each chunk, and compare the hash for each chunk with stored MD5 hashes
  - Only updates chunks with changed hashes



Incremental backup

- Stores all changes between current time and last backup (not necessarily full backup)
- Smallest storage space
- Hard to recover (if full backup was a long time ago)
- What happens if we combine differential and incremental backups?



Replication

- Different from backups: replication keeps no historical state
- Synchronous replication: All file updates should happen (almost) immediately
- Asynchronous replication: Small delay when pushing to replicas is acceptable

Data Privacy

Data has sensitive attributes and personally identifiable information

How can the data owner allow a data user to utilize the data without compromising privacy?

Idea: Restrict queries by data user
But this leads to *inference attacks!*

Data Privacy

How can the data user compute Q on data owner's D without compromising privacy?

- <u>k-Anonymity</u>: (D sensitive, Q possibly sensitive) Publish distorted D
- <u>Differential Privacy</u>: (D sensitive) Allow only special queries with mathematical error guarantees
- <u>Secure Multiparty Computation</u>: (D1, D2 sensitive) jointly compute Q without revealing D1, D2 to each other
- <u>Private Information Retrieval</u>: (Q sensitive) Retrieve some information from D without revealing Q

k-Anonymity

- Remove link between identifiers (PII) and sensitive attribute
- Anonymization function is usually deterministic
- After anonymization, each **set** of identifiers in the table must appear at least *k* times (= anonymity sets have at least k elements)

(data points)

k-Anonymity

	Quasi-identifiers		
	Age	Weight (kg)	Heart disease?
Hospital Subjects	23	86	N
	15	65	Y
	34	123	Y
	55	95	N
	32	63	Y
	45	89	Y
	59	112	N
	61	81	Y
	15	73	Y

→

	Quasi-identifiers		
	Age	Weight (kg)	Heart disease?
Hospital Subjects	25	100	N
	25	50	Y
	25	100	Y
	50	100	N
	25	50	Y
	50	100	Y
	50	100	N
	50	100	Y
	25	50	Y

"Round Age to nearest 25, Weight to nearest 50" → k = 2
(There are three anonymity sets: Size 2, Size 3, Size 4.
We take the minimum to be k.)

k-Anonymity

	Quasi-identifiers		
	Age	Weight (kg)	Heart disease?
Hospital Subjects	25	100	N
	25	50	Y
	25	100	Y
	50	100	N
	25	50	Y
	50	100	Y
	50	100	N
	50	100	Y
	25	50	Y

- A flaw in k-anonymity: All members of an anonymity set may have the same sensitive attribute
  - e.g. If your friend is around age 25 and weight 50kg, and you know they're in the table, you know they have heart disease
- To fix this, we can also enforce *l*-diversity: Every anonymity set must have at least *l* different sensitive attributes

k-Anonymity

- Another weakness is that complementary releases can compromise k-anonymity:

	Quasi-identifiers	
	Weight (kg)	Sickness
Hospital Subjects	30-60	A
	30-60	B
	30-60	C
	30-60	D
	30-60	E
	60-150	F
	60-150	G
	60-150	H
	60-150	I

$k = 4$

	Quasi-identifiers	
	Weight (kg)	Sickness
Hospital Subjects	25-55	A
	25-55	B
	25-55	C
	25-55	D
	55-150	E
	55-150	F
	55-150	G
	55-150	H
	55-150	I

$k = 4$

Different rounding schemes will weaken k - if your friend has weight 56 kg, you know they have E

k-Anonymity

- Knowing the anonymization scheme can also compromise the scheme. Suppose Age is the only QID. If you know the anonymization scheme is the following:
  - Sort patients by age, start with an anonymity set containing only the smallest age.
  - Add patients in order to the current anonymity set until desired *k* and *l* have been achieved. Then start a new anonymity set with the next person that has not been added.
  - Repeat until all patients added; if final anonymity set is too small, merge it with the previous completed anonymity set.
- Suppose the hospital want to achieve $k = 3$, $l = 2$. It releases two anonymity sets {Age}:{Heart Disease} as follows:

  - {0-40}: {N, Y, Y, Y, Y}   {40-80}: {Y, N N}
- If you know that your friend is the youngest person in the database, then they definitely have heart disease, otherwise the first set would not be so large!

Differential privacy

- Ensures privacy of data items using a *differential* mathematical formulation
- Hard to understand, easy to implement
- Anonymization function is random
- Used in iOS 10 (2016)
- Two uses: prevent data queries from compromising privacy, and allow data collection

Differential privacy

Two databases are *neighboring* if they are the same except for one element (one person's data).

A query Q is *ε-differentially private if for all neighboring databases D_1 and D_2 and for all q:*

$$\frac{Pr(Q(D_1)=q)}{Pr(Q(D_2)=q)} \leq e^{\varepsilon}$$

Intuitively, changing one person's data is unlikely to change the result (distribution) of a differentially private query => the query result does not reveal that person's existence!

Differential privacy

Suppose the salaries of 5 employees are:

Employee	Salary
A	$200
B	$210
C	$240
D	$150
E	$400

For legal compliance, the company is obligated to reveal the average salary of its employees.

Now E has left the company. Suppose someone queries the average salary twice: before and after E left. The results are: $200, $240. This reveals E's salary: $240 * 5 - $200 * 4 = $400.

Differential privacy

<u>Differential privacy:</u> Implicitly add noise before returning the average.

- This data is never revealed, noise is only added once, “internally”

Employee	Salary
A	$200+Noise
B	$210+Noise
C	$240+Noise
D	$150+Noise
E	$400+Noise

This satisfies differential privacy, generally:

- Difference between two neighboring data sets is dominated by noise
- In other words, one person joining/leaving doesn't change the result as much as the noise itself
- The mean has less noise than any other element

Satisfying differential privacy

Does Laplace distribution noise satisfy differential privacy for mean query?

Suppose two neighboring datasets D_1 and D_2 have means M_1 and M_2

We write $k = M_1 + x_1 = M_2 + x_2$

$$\frac{Pr(Q(D_2)=k)}{Pr(Q(D_1)=k)}=\frac{e^{-|x_2|/b}}{e^{-|x_1|/b}}\leqslant e^{|x_1-x_2|/b}=e^{|M_1-M_2|/b}$$

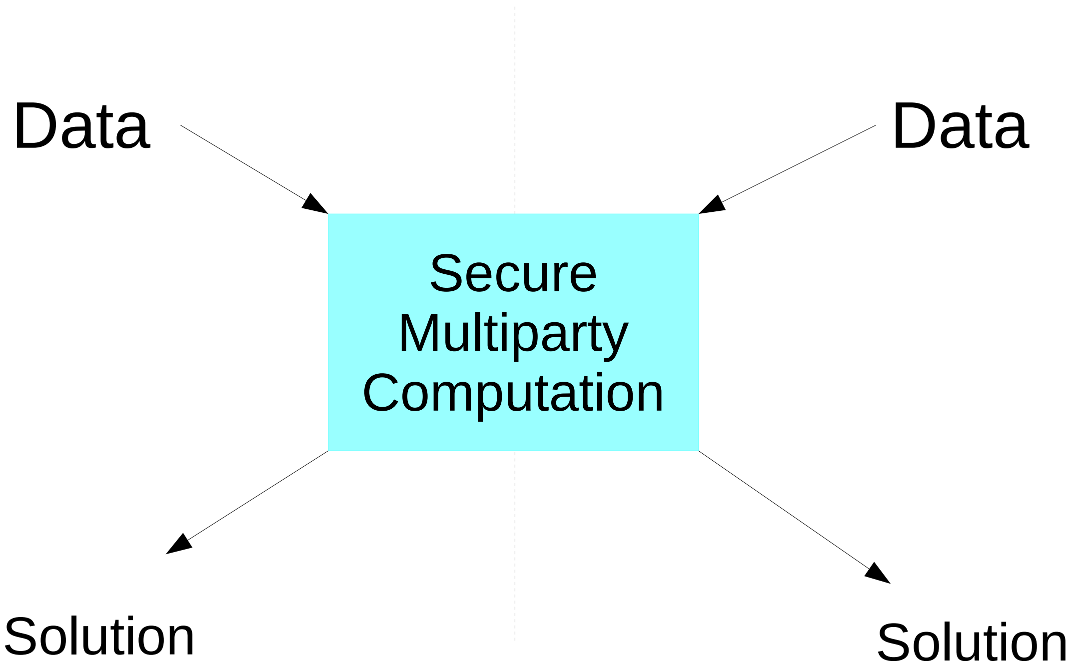
If we can bound $|M_1-M_2|$, which is the maximum difference between two query results, we achieve ($|M_1-M_2|/b$)-differential privacy

Note that normal distribution noise does not satisfy differential privacy in general

Differential privacy

- Note the proof applies to any function (replace M_1, M_2 with $f(D_1)$, $f(D_2)$)
  - $\max(|f(D_1) - f(D_2)|)$ is known as the "sensitivity"
  - Functions with good (low) sensitivity: COUNT, SUM
  - Functions with bad sensitivity: MAX, MIN, Median
- To avoid multiple queries reducing the noise level of the data, we sample the noise only once until the data changes
- Can be applied to many types of data/queries
- Useful for data collection – each individual adds large noise before sending to data collector ("query" is invisible)

Secure Multiparty Computation



Useful for research, data analytics, collaboration, etc.

Secure Multiparty Computation

- Two parties with different data can jointly compute a known function on the union of their data while sharing no data at all
  - e.g. “Who has more customers on this day?”
- Generally (much) slower than directly running the algorithm
  - e.g. 20 minutes on 2 cores to complete one AES encryption of 128 bits under SMPC
- Guaranteed correctness without noise

A different scenario

- What if the data holder's data is not sensitive, but the data user's query is sensitive?
- For example:
  - Searching for a patent
  - Searching for attributes of a sickness
  - Searching for darknet sites
- We want to use Private Information Retrieval in these cases

Private Information Retrieval

- Sometimes we want to query a database without revealing the query to the database
  - e.g. Asking a medical database about your symptoms
  - e.g. Asking a patent database about a new idea
- Trivial but impractical solution: Download the entire database. To achieve practical solutions we need to reduce communication overhead
- Two types:
  - Information-theoretic PIR: Several databases will each obtain a “portion” of the query
    - No knowledge of the real query is leaked
    - Impossible for single database
  - Computational PIR: Databases need to compute difficult problem to obtain query