CS 458 / 658
Computer Security and Privacy

Module 6
Database Security and Privacy

Spring 2013

Module outline

1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

Module outline

1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

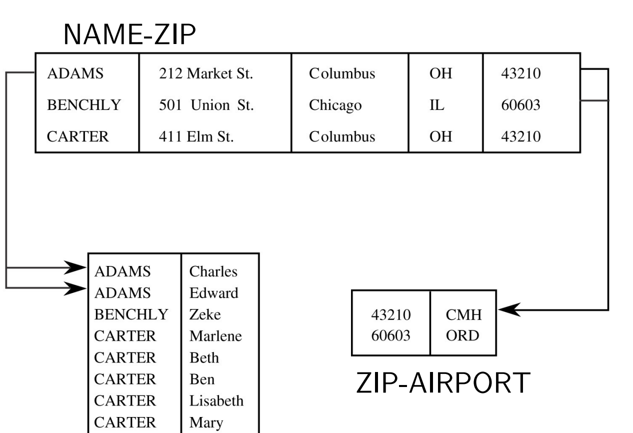
(Relational) Databases

- Structured, queryable collection of data (records)
- Each record consists of fields (elements)
- Structure (schema) set by database administrator
- Database management system (DBMS) provides support for queries and management
- Most popular DBMS is based on relational model
- Stores records in one or multiple tables (relations)
  - Table has named columns (attributes) and rows (tuples)
  - Individual tables can have relationships between them

Records

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

Relations



(attributes are missing)

Database queries

- Most popular query language is SQL
  - `SELECT Address FROM NAME-ZIP`
    `WHERE (Zip = '43210') AND (Name = 'ADAMS')`
    - Prints address of family in relation NAME-ZIP whose zip code is 43210 and whose name is Adams
  - `SELECT Name, Airport`
    `FROM NAME-ZIP, ZIP-AIRPORT`
    `WHERE NAME-ZIP.Zip = ZIP-AIRPORT.Zip`
    - Prints each person's last name and his/her airport by joining relations NAME-ZIP and ZIP-AIRPORT
  - `SELECT COUNT(Name) FROM NAME-ZIP`
    `WHERE City = 'Columbus'`
    - Prints number of families in Columbus
    - Can also do other computations, like `SUM`, `MIN`, or `AVG`
- Result of a query is a subschema

Module outline

1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

Security requirements

- Physical database integrity
- Logical database integrity
- Element integrity
- Referential integrity
- Auditability
- Access control
- User authentication
- Availability

Database integrity

- Logical and physical integrity
- Protect against database corruption
  - Allow only authorized individuals to perform updates
- Recover from physical problems (Power failures, disk crashes,. . . .)
  - Perform periodic backups
  - Keep log of transactions to replay transactions since last backup

Element integrity

- Ensure correctness/accuracy of database elements
- Access control to limit who can update element
- Element checks to validate correctness
  - Element must be numeric, within a particular range,. . .
  - Not more than one employee can be president
  - Helps against mistakes by authorized users
  - Typically enforced by triggers (procedures that are automatically executed after an `INSERT`, `DELETE`,. . .)

Element integrity (cont.)

- Change log or shadow fields to undo erroneous changes
  - In case access control or element checks fail
  - Require additional space in the database
- Error detection codes to protect against OS or hard disk problems

Integrity: two-phase update

- For a set of operations, either all of them or none of them should be performed
  - Integrity violation if only some are performed
  - E.g., money is withdrawn from an account, but not deposited to another account
- First phase: gather information required for changes, but don't perform any updates, repeat if problem arises (shadow fields)
- Second phase: make changes permanent, repeat if problem arises
- See text for example

Integrity: concurrency control

- Concurrent modifications can lead to integrity violation
  - Two operations A and B read variable X
  - A then writes new value of X
  - B then writes new value of X
  - A's update gets lost
- Need to perform A and B as atomic operations
- Take CS 454 for more about this

Referential integrity

- Each table has a primary key, which is a minimal set of attributes that uniquely identifies each tuple
  - User ID or social insurance number
  - First name and last name (maybe not)
- A table might also have a or multiple foreign keys, which are primary keys in some other table
  - Zip is (likely) a primary key in ZIP-AIRPORT
  - Zip is a foreign key in NAME-ZIP
- Referential integrity ensures that there are no dangling foreign keys
  - For each zip in NAME-ZIP, there is an entry in ZIP-AIRPORT

Auditability

- Keep an audit log of all database accesses
  - Both read and write
- Access control can be difficult (see later), audit log allows to retroactively identify users who accessed forbidden data
  - Police officer looking at somebody's criminal record as a favor to a friend, unauthorized medical personnel looking at Britney Spears' medical records
- Maybe combination of accesses resulted in disclosure, not a single one (see later)
- Must decide about granularity of logging
  - Should results of a query be logged?

Access control

- More difficult than OS access control
- Might have to control access at the relation, record or even element level
- Many types of operations, not just read/write
  - `SELECT, INSERT, UPDATE, CREATE, DROP,...`
- Relationships between database objects make it possible to learn sensitive information without directly accessing it
  - Inference problem (see later)
- Efficiency problem in presence of thousands of records, each consisting of dozens of elements

Access control (cont.)

- Access control might consider past queries
  - Current query, together with past ones, could reveal sensitive information
    - Iteratively querying whether element is in set ultimately leaks set
- Or type of query
  - ```
    SELECT lastname, salary FROM staff
     WHERE salary > 50000
    ```
    might be forbidden, but not
  - ```
    SELECT lastname FROM staff
     WHERE salary > 50000
    ```

User authentication / Availability

- Database might do its own authentication
- Additional checks possible
  - E.g., time of day
- Databases facilitate sharing, but availability can suffer if multiple users want to access the same record
  - Block access until other user finishes updating record

Module outline

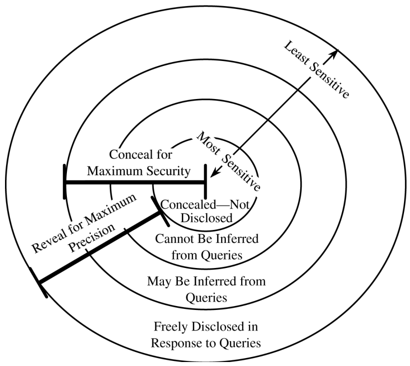
1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

Types of data disclosure

- Exact data
- Bounds
  - Sensitive value is smaller than H, but bigger than L
  - Might iteratively decrease range (binary search)
- Negative result
  - Knowing that a person does not have zero felony convictions is sensitive, even if actual number is hidden
- Existence
  - Knowing of existence of some data can be sensitive
- Probable value
  - Sensitive data has value x with probability y

Security vs. precision

- Security: Forbid any queries that access sensitive data, even if (aggregated) result is no longer sensitive
- Precision: Aggregated result should reveal as much non-sensitive data as possible



Data inference

- Derivation of sensitive data from (supposedly) non-sensitive data
- Direct attack
  - Attacker issues query that directly yields sensitive data
  - Might obfuscate query to fool DBMS
    - `SELECT SUM(salary) FROM staff`
      `WHERE lastname = 'Adams'`
      `OR (sex != 'M' AND sex != 'F')`
- Indirect attack
  - Infer sensitive data from statistical results
    - As released by governments or pollers
  - Tracker attack

Statistical inference attacks

- Sum
  - Leaks sensitive data if sum covers only one record or if attacker can control set of covered records
    - `SELECT SUM(salary)`
    - `SELECT SUM(salary) WHERE lastname != 'Adams'`
- Count
  - Useful in attack above
- Mean
  - sum = count * mean
- Median
  - Intersecting medians might leak sensitive data
  - See text for example

Tracker attacks

- Assume that there is a query C that DBMS refuses to answer since it matches fewer than k or more than $N-k$ (but fewer than N) records
  - N: number of records in database
  - Why the more than $N-k$ restriction?
- A tracker T is a query whose result matches between $2k$ and $N-2k$ records
  - DBMS will answer query T and the query *not* T

Tracker attacks (cont.)

- Let $q()$ be the result of a query (e.g., a COUNT or SUM query) and let S be the set of all records
- Using Venn diagrams, we can show that
  - $q(C) = q(C \text{ or } T) + q(C \text{ or not } T) - q(S)$
  - Use right-hand side for computing $q(C)$ if $q(C)$ matches fewer than k records
  - $q(C) = 2 * q(S) - q(\text{not } C \text{ or } T) - q(\text{not } C \text{ or not } T)$
  - Use right-hand side for computing $q(C)$ if $q(C)$ matches more than $N-k$ records
- In general, simple logic or linear algebra might allow an attacker to convert a forbidden query into multiple, allowed queries

Controls for statistical inference attacks

- Apply control to query or to data items
  - As seen, former is difficult
- Suppression and concealing are two controls applied to data items
- Suppression
  - Suppress sensitive data from result
- Concealing
  - Answer is close to actual value, but not exactly

Controls (cont.)

- n-item k-percent rule
  - For the set of records that were included in the result, if there is a subset of n records that is responsible for over k percent of the result, omit the n records from result
  - However, omission itself might leak information or omitted value could be derived with other means
- Combined results
  - Report set or range of possible values
- Random sample
  - Compute result on random sample of database
  - Need to use same sample for equivalent queries

Controls (cont.)

- Random data perturbation
  - Add or subtract small random error to/from each value before computing result
  - Expectation is that statistical properties are maintained
- Query analysis
  - Maintain history of user's queries and observe possible inferences
  - Costly, fails for colluding users

Differential Privacy

- The response to a query should not depend on an individual (not) being part of the dataset
- A query K has ϵ-differential privacy if for all datasets D and D', where D and D' differ in at most one row, the probability that $K(D)$ has a particular output is at most e^{ϵ} * the probability that $K(D')$ has this output $(0 \leq \epsilon \leq 1)$
- Typically differential privacy is achieved by adding noise to the result of a query before releasing it
- Differential privacy is an active topic of research and has been incorporated into MapReduce and SQL databases

Data aggregation

- Data aggregation is related to data inference
- Building sensitive results from less sensitive inputs
- Aggregation can take place outside of a DBMS, which makes it difficult to control
  - People with different access rights talking to each other
- Closely related to data mining (see later), where information from different databases is combined

Module outline

1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

Multilevel Security (MLS) Databases

- Support classification/compartmentalization of information according to its confidentiality
  - E.g., two sensitivity levels (sensitive and not sensitive)
- At element level if necessary
  - Salary might be sensitive only for some employees
  - Other information in employee's record might not be sensitive
- In an MLS database, each object has a sensitivity classification and maybe a set of compartments
  - Object can be element, aggregate, column, or row

*-Property

- Implementing the *-property (no read up, no write down) in an MLS database is difficult
  - User doing a write-up, even though the user cannot read the data having higher sensitivity (Blind writes)
  - Write-downs need a sanitization mechanism
  - Trusted processes that can do anything
- DBMS must have read and write access at all levels to answer user queries, perform back-ups, optimize database,...
  - Must trust DBMS

Confidentiality

- Depending on a user's clearance, he/she might get different answers for a query
  - Less precision for low-clearance users
- Existence of a record itself could be confidential
- Keeping existence hidden can lead to having multiple records with the same primary key, but different sensitivity (polyinstantiation)
  - Admin notices that there is no record for employee Bob Hill and creates one
  - However, Bob Hill is a secret agent, so there already is a record, which admin cannot see
  - DBMS must allow admin's request, else admin would get suspicious

Partitioning

- Have separate database for each classification level
- Simple, often used in practice
- Might lead to data stored redundantly in multiple databases
- Doesn't address the problem of a high-level user needing access to low-level data combined with high-level data

Encryption

- Separate data by encrypting it with a key unique to its classification level
- Must be careful to use encryption scheme in the right way
  - E.g., encrypting the same value in different records with the same key should lead to different ciphertexts
- Processing of a query becomes expensive, many records might have to be decrypted
  - Doing the processing directly on the encrypted data is an active research area (homomorphic encryption)

Integrity lock

- Provides both integrity and access control
- Each data item consists of
  - The actual data item
  - An integrity level (maybe concealed)
  - A cryptographic signature (or MAC) covering the above plus the item's attribute name and its record number
- Signature protects against attacks on the above fields, such as attacks trying to modify the sensitivity label, and attacks trying to move/copy the item in the database
- This scheme does not protect against replay attacks

Integrity lock (cont.)

- Any (untrusted) database can be used to store data items and their integrity locks
  - Locks can consume lots of space (maybe multiple locks per record)
- (Trusted) procedure handles access control and manages integrity locks
  - E.g., updates integrity level to enforce *-property or re-computes signature after a write access
  - Expensive
- Have to encrypt items and locks if there are other ways to get access to data in database
  - Makes query processing even more expensive

Module outline

1 Introduction to databases

2 Security requirements

3 Data disclosure and inference

4 Multilevel security databases

5 Designs of secure databases

6 Data mining and data release

Trusted front end

- Front end authenticates a user and forwards user query to old-style DBMS
- Front end gets result from DBMS and removes data items that user is not allowed to see
- Allows use of existing DBMS and databases
- Inefficient if DBMS returns lots of items and most of them are being dropped by front end

Commutative filters

- Front end re-writes user query according to a user's classification
  - Remove attributes that user is not allowed to see
  - Add constraint expressing user's classification
- Benefits from DBMS' superior query processing capabilities and discards forbidden data items early on
- Front end might still have to do some post processing

Distributed/federated databases

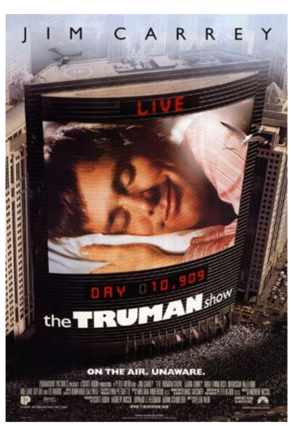
- Based on partitioning
- Front end forwards user query only to databases that user can access based on classification
- Front end might have to combine the results from multiple databases
  - Complex process, front end essentially becomes a DBMS
- Doesn't scale to lots of classification levels

Views

- Many DBMS support views
- A view is logical database that represents a subset of some other database
  - `CREATE VIEW foo AS SELECT * FROM bar WHERE...`
- Element in view can correspond to an element in underlying database or be a combination of multiple elements
  - E.g., their sum
- Views can be used for access control
  - A user's view of a database consists of only the data that the user is allowed to access
  - Hide attribute/row unless user is allowed to access at least one element, set to UNDEFINED any elements that user can't access

Truman vs. non-Truman semantics

- Truman semantics: the DBMS pretends that the data the user is allowed to access is all the data there is
  - Like "The Truman Show"
  - All queries will succeed, even if they return incorrect results
- Non-Truman semantics: the DBMS can reject queries that ask for data the user is not allowed to access
  - Any queries that succeed will produce precise answers
  - Some queries will fail



Module outline

1. Introduction to databases
2. Security requirements
3. Data disclosure and inference
4. Multilevel security databases
5. Designs of secure databases
6. Data mining and data release

Data mining

- Multilevel databases weren't a commercial success
  - Mainly military clients, finding all possible inferences is NP-complete
- However, the combination of (sensitive) information, stored in multiple (maybe huge) databases, as done for data mining, raises similar concerns and has gotten lots of attention recently
- So far, a single entity has been in control of some data
  - Knows what kind of data is available
  - Who has accessed it (ignoring side channels)
- No longer the case in data mining, data miners actively gather additional data from third parties

Data mining (cont.)

- Data mining tries to automatically find interesting patterns in data using a plethora of technologies
  - Statistics, machine learning, pattern recognition,. . .
  - Still need human to judge whether pattern makes sense (causality vs. coincidence)
- Data mining can be useful for security purposes
  - Learning information about an intrusion from logs

Security problems of data mining

- Confidentiality
  - Derivation of sensitive information
- Integrity
  - Mistakes in data
- Availability
  - (In)compatibility of different databases

Confidentiality

- Data mining can reveal sensitive information about humans (see later) and companies
- In 2000, the U.S. National Highway Traffic Safety Administration combined data about Ford vehicles with data about Firestone tires and become aware of a problem with the Ford Explorer and its Firestone tires
  - Problem started to occur in 1995, and each company individually had some evidence of the problem
  - However, data about product quality is sensitive, which makes sharing it with other companies difficult
- Supermarket can use loyalty cards to learn who buys what kind of products and sell this data, maybe to manufacturers' competitors

Data correctness and integrity

- Data in a database might be wrong
  - E.g., input or translation errors
- Mistakes in data can lead to wrong conclusions by data miners, which can negatively impact individuals
  - From receiving irrelevant mail to being denied to fly
- Privacy calls for the right of individuals to correct mistakes in stored data about them
  - However, this is difficult if data is shared widely or if there is no formal procedure for making corrections
- In addition to false positives, there can also be false negatives: don't blindly trust data mining applications

Availability

- Mined databases are often created by different organizations
  - Different primary keys, different attribute semantics,. . .
    - Is attribute "name" last name, first name, or both?
    - US or Canadian dollars?
- Makes combination of databases difficult
- Must distinguish between inability to combine data and inability to find correlation

Privacy and data mining

- Data mining might reveal sensitive information about individuals, based on the aggregation and inference techniques discussed earlier
- Avoiding these privacy violations is active research
- Data collection and mining is done by private companies
  - Privacy laws (e.g., Canada's PIPEDA or U.S.' HIPAA) control collection, use, and disclosure of this data
  - Together with PETs
- But also by governments
  - Programs tend to be secretive, no clear procedures
  - Phone tapping in U.S., no-fly lists in U.S. and Canada

Privacy-preserving data release

- Anonymize data records before releasing them
  - E.g., strip names, addresses, phone numbers
  - Unfortunately, such simple anonymization might not be sufficient
- August 6, 2006: AOL released 20 million search queries from 658,000 users
- To protect users' anonymity, AOL assigned a random number to each user
  - 4417749 "numb fingers"
  - 4417749 "landscapers in Lilburn, GA"
  - 17556639 "how to kill your wife"
- August 9: New York Times article re-identified user 4417749
  - Thelma Arnold, 62-year old widow from Lilburn, GA

Another example (by L. Sweeney)

- 87% of U.S. population can be uniquely identified based on person's ZIP code, gender, and date of birth
- Massachusetts' Group Insurance Commission released anonymized health records
- Records omitted individuals' names, but gave their ZIP codes, gender, and date of birth (and health information, of course)
- Massachusetts's voter registration lists contain these three items plus individuals' names and are publicly available
- Enables re-identification by linking

k-anonymity [2002]

- Ensure that for each released record, there are at least $k-1$ other released records from which record cannot be distinguished (where $k \geq 2$)
- For health-records example, release a record only if there are $k-1$ other records that have same ZIP code, gender, and date of birth
  - Assumption: there is only one record for each individual
- Because of the 87% number, this won't return many records, need some pre-processing of records
  - Remove ZIP code, gender, or date of birth
  - Reduce granularity of ZIP code or date of birth (domain generalization)

Discussion

- In health-records example, the attributes ZIP code, gender, and date of birth form a "quasi-identifier"
- Determining which attributes are part of the quasi-identifier can be difficult
  - Should health information be part of it?
  - Some diseases are rare and could be used for re-identification
- Quasi-identifier should be chosen such that released records do not allow any re-identification based on any additional data that attacker might have
  - Clearly we don't know all this data

Limitations of k-anonymity

A 3-anonymized table

ZIP	DOB	Disease
902**	196*-*-*	Cancer
902**	196*-*-*	Cancer
902**	196*-*-*	Cancer
902**	195*-*-*	Heart disease
902**	195*-*-*	GI disease
902**	195*-*-*	Flu
904**	195*-*-*	Heart disease
904**	195*-*-*	Cancer
904**	195*-*-*	Cancer

ℓ-diversity and t-closeness

- Homogeneity attack
  - If you know Bob (902**,196*-*-*) is in the table, then Bob has cancer.
- Background knowledge attack
  - If you know Dave (904**,195*-*-*) is in the table, and that his risk for heart disease is very low, then Dave has cancer.
- ℓ-diversity property [2006]:
  - For any quasi-identifier, there should be at least ℓ "well-represented" values of the sensitive fields
- Possibly still not good enough: t-closeness [2007]
  - Ensure that the distributions of the values for any quasi-identifier are within t of the distribution for the whole table

$\Rightarrow$ Active research area

Value swapping

- Data perturbation based on swapping values of some (not all!) data fields for a subset of the released records
  - E.g., swap addresses in subset of records
- Any linking done on the released records can no longer considered to be necessarily true
- Trade off between privacy and accuracy
- Statistically speaking, value swapping will make strong correlations less strong and weak correlations might go away entirely

Adding noise

- Data perturbation based on adding small positive or negative error to each value
- Given distribution of data after perturbation and the distribution of added errors, distribution of underlying data can be determined
  - But not its actual values
- Protects privacy without sacrificing accuracy

Sampling / Synthetic data

- Release only a subset of respondents' data (e.g., a 1% sample) with geographic coarsening and top/bottom coding
  - Geographic coarsening: restrict geographic identifiers to regions containing at least a certain population (e.g., 100,000 people)
  - Top/bottom-coding: for example, if there are sufficiently few respondents over age 90, top-coding would replace all ages ≥ 90 with the value 90
- Build a distribution model based on gathered data and use the model to generate synthetic data with similar characteristics to original data
  - Release one (or a few) sets of synthetic data

Recap

- Introduction to databases
- Security requirements
- Data disclosure and inference
- Multilevel security databases
- Designs of secure databases
- Data mining and data release