DATABASE SECURITY

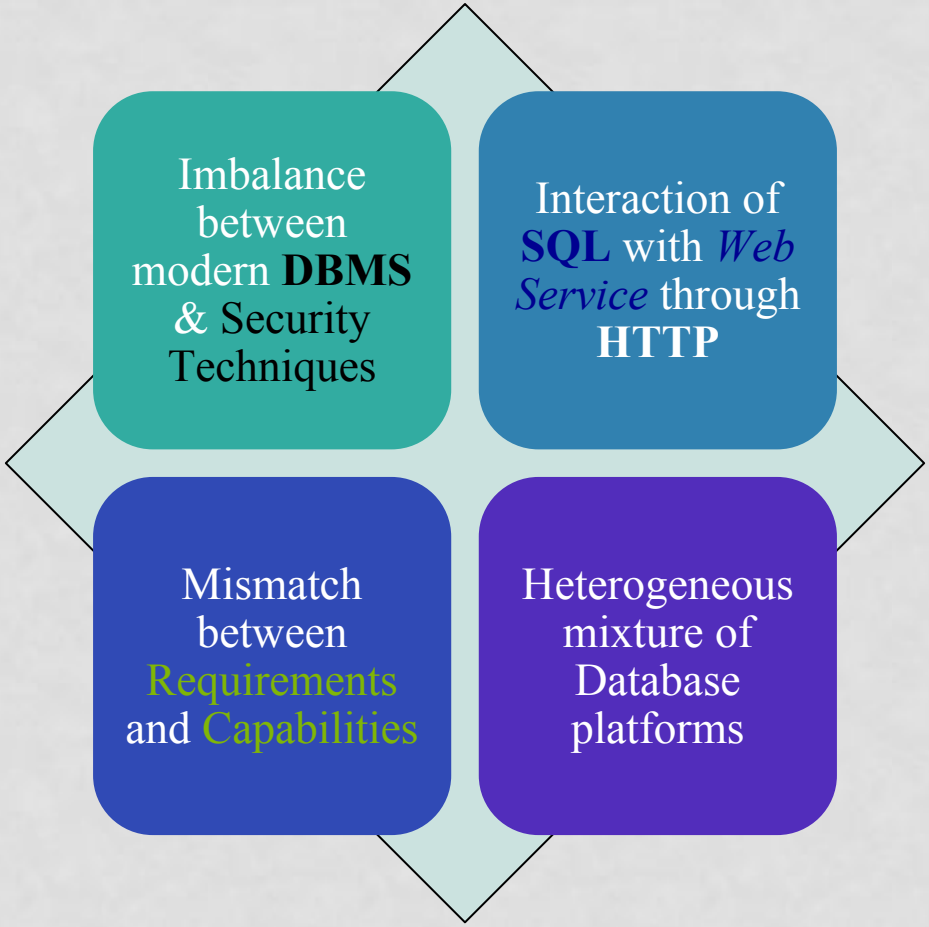
MD. IFTEKHARUL ALAM EFAT

Institute of Information Technology
Noakhali Science & Technology University

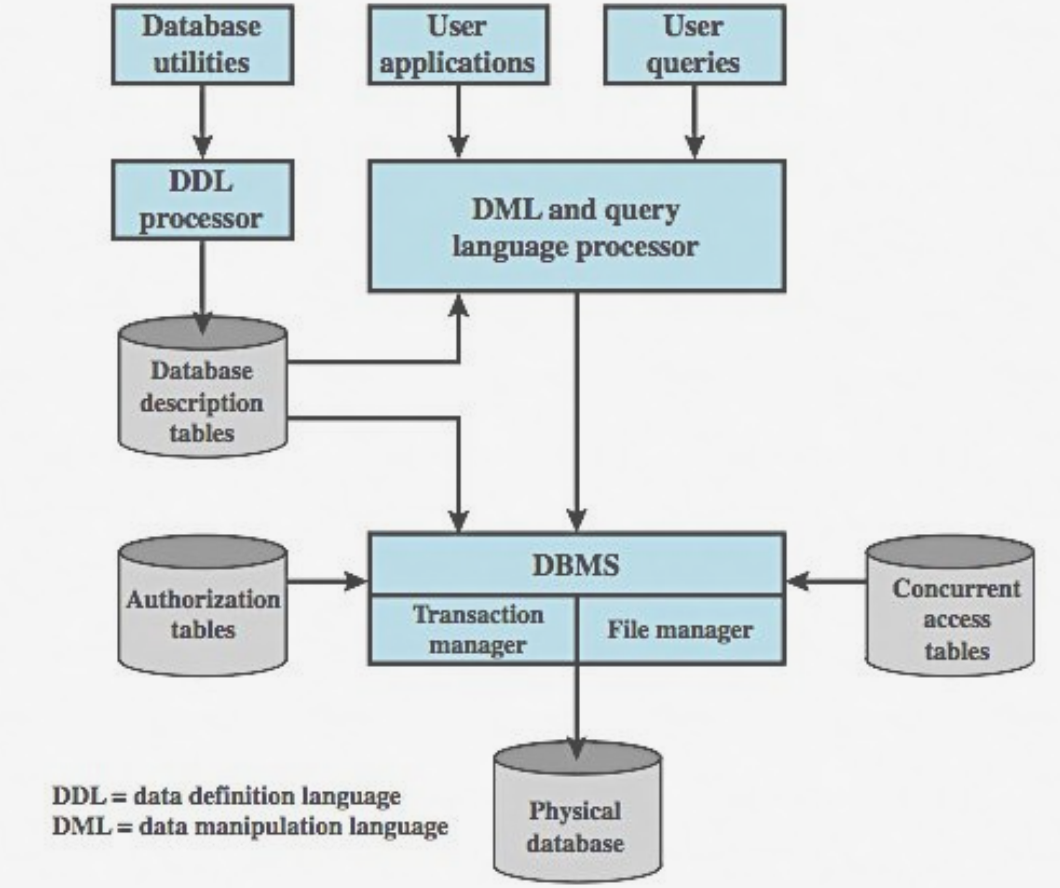
LEARNING OBJECTIVES

- Understand the unique need for database security, separate from ordinary computer security measures
- Present an overview of the basic elements of a database management system
- Present an overview of the basic elements of a relational database system
- Define and explain SQL injection attacks
- Compare and contrast different approaches to database access control
- Explain how inference poses a security threat in database systems
- Discuss the use of encryption in a database system
- Present an overview of cloud computing concepts
- Understand the unique security issues related to cloud computing

THE NEED


Imbalance between modern **DBMS** & Security Techniques
Interaction of **SQL** with *Web Service* through **HTTP**
Mismatch between Requirements and Capabilities
Heterogeneous mixture of Database platforms

DBMS ARCHITECTURE

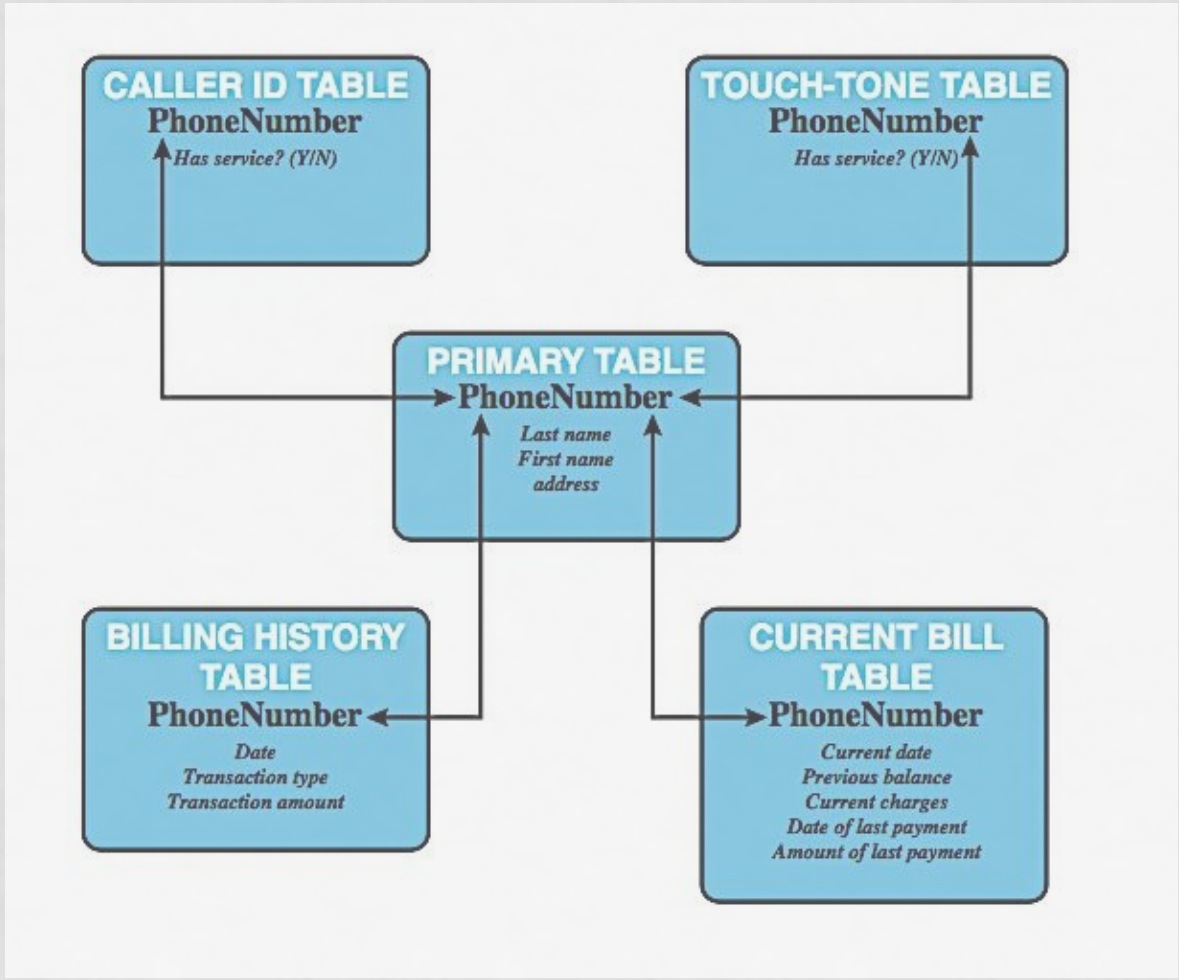


RELATIONAL DATABASES

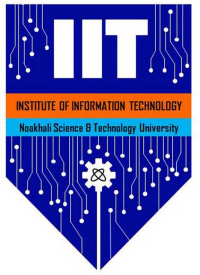
- Constructed from tables of data
  - Each column holds a particular type of data
  - Each row contains a specific value these
  - Ideally has one column where all values are unique, forming an identifier/key for that row
- Have multiple tables linked by identifiers
- Use a query language to access data items meeting specified criteria

EXAMPLE OF RDBMS

A relational database uses multiple tables related to one another by a designated key; in this case the key is the Phone-Number field.



RELATIONAL DATABASE ELEMENTS



- Primary key
  - Uniquely identifies a row
- Foreign key
  - Links one table to attributes in another
- View / virtual table

Formal Name	Common Name	Also Known As
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

RELATIONAL DATABASE ELEMENTS

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

primary key

Employee Table

Ename	Did	SalaryCode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key — primary key

(a) Two tables in a relational database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

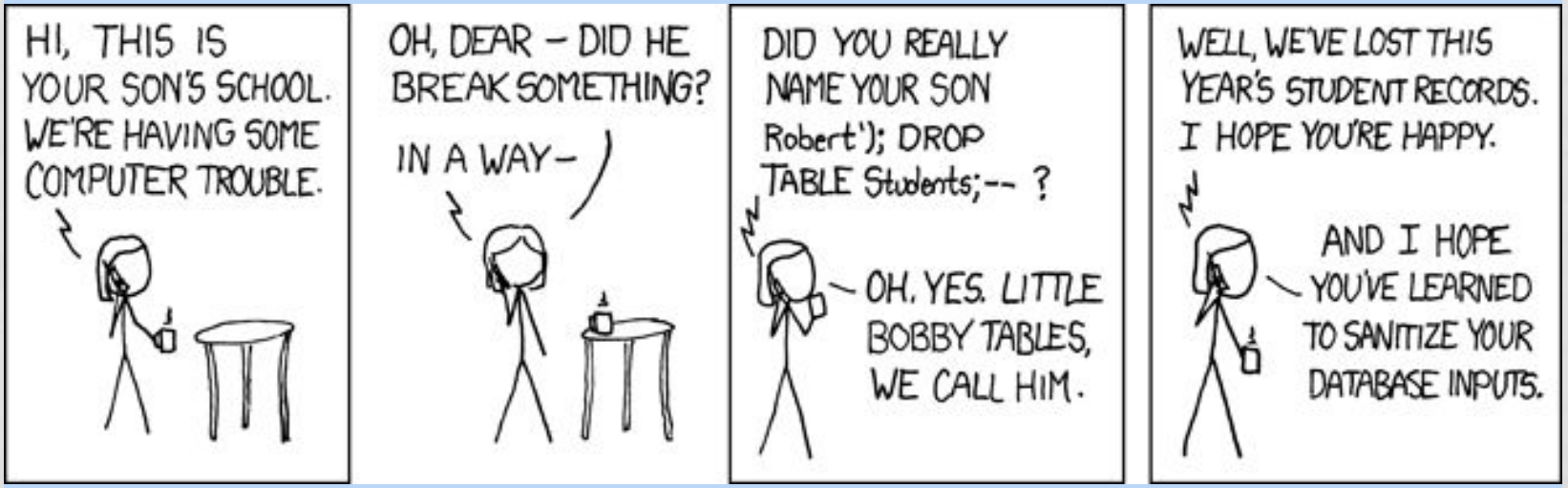
STRUCTURED QUERY LANGUAGE

- Structure query language (SQL)
  - Originally developed by IBM in the mid-1970s
  - Standardized language to define, manipulate, and query data in a relational database
  - Several similar versions of ANSI/ISO standard

```
CREATE TABLE department (
     Did INTEGER PRIMARY KEY,
     Dname CHAR (30),
     Dacctno CHAR (6) )

CREATE TABLE employee (
     Ename CHAR (30),
     Did INTEGER,
     SalaryCode INTEGER,
     Eid INTEGER PRIMARY KEY,
     Ephone CHAR (10),
     FOREIGN KEY (Did) REFERENCES department (Did) )
```

```
CREATE VIEW newtable (Dname, Ename, Eid, Ephone)
AS SELECT D.Dname E.Ename, E.Eid, E.Ephone
FROM Department D Employee E
WHERE E.Did = D.Did
```



SQL INJECTION ATTACK

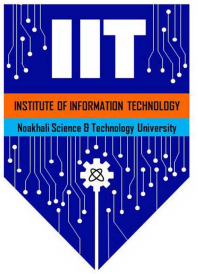

KuvatON.com


DW 530GS
ZWOLNIJ


ZWOLNIJ


ZU 0666', 0, 0); DROP DATABASE TABLICE

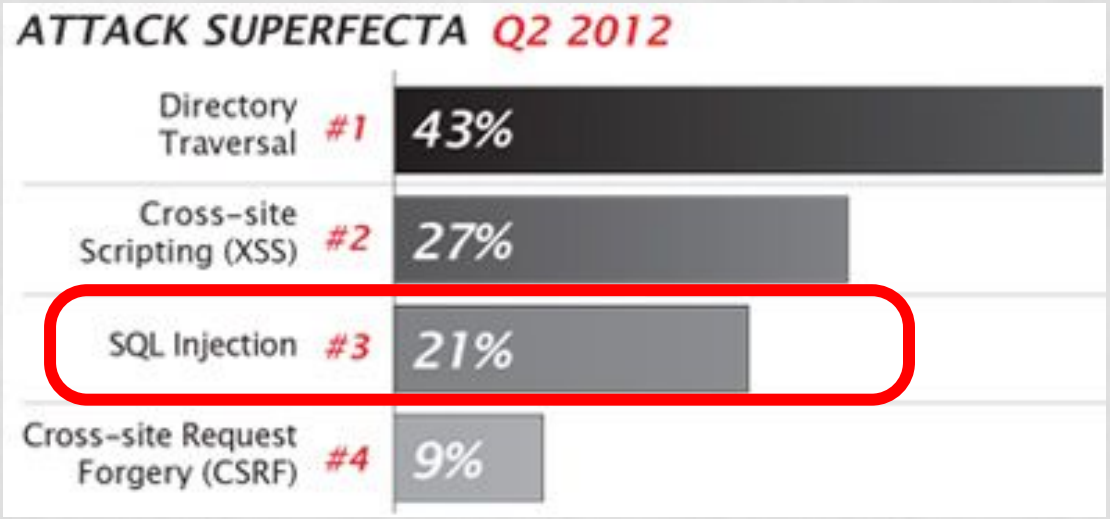
WHAT IS A SQL INJECTION ATTACK?



- Many web applications take user input from a form
- Often this user input is used literally in the construction of a SQL query submitted to a database. For example:
  - `SELECT productdata FROM table WHERE productname = 'user input product name;`
- A SQL injection attack involves placing SQL statements in the user input

SQL INJECTION ATTACKS ON THE RISE

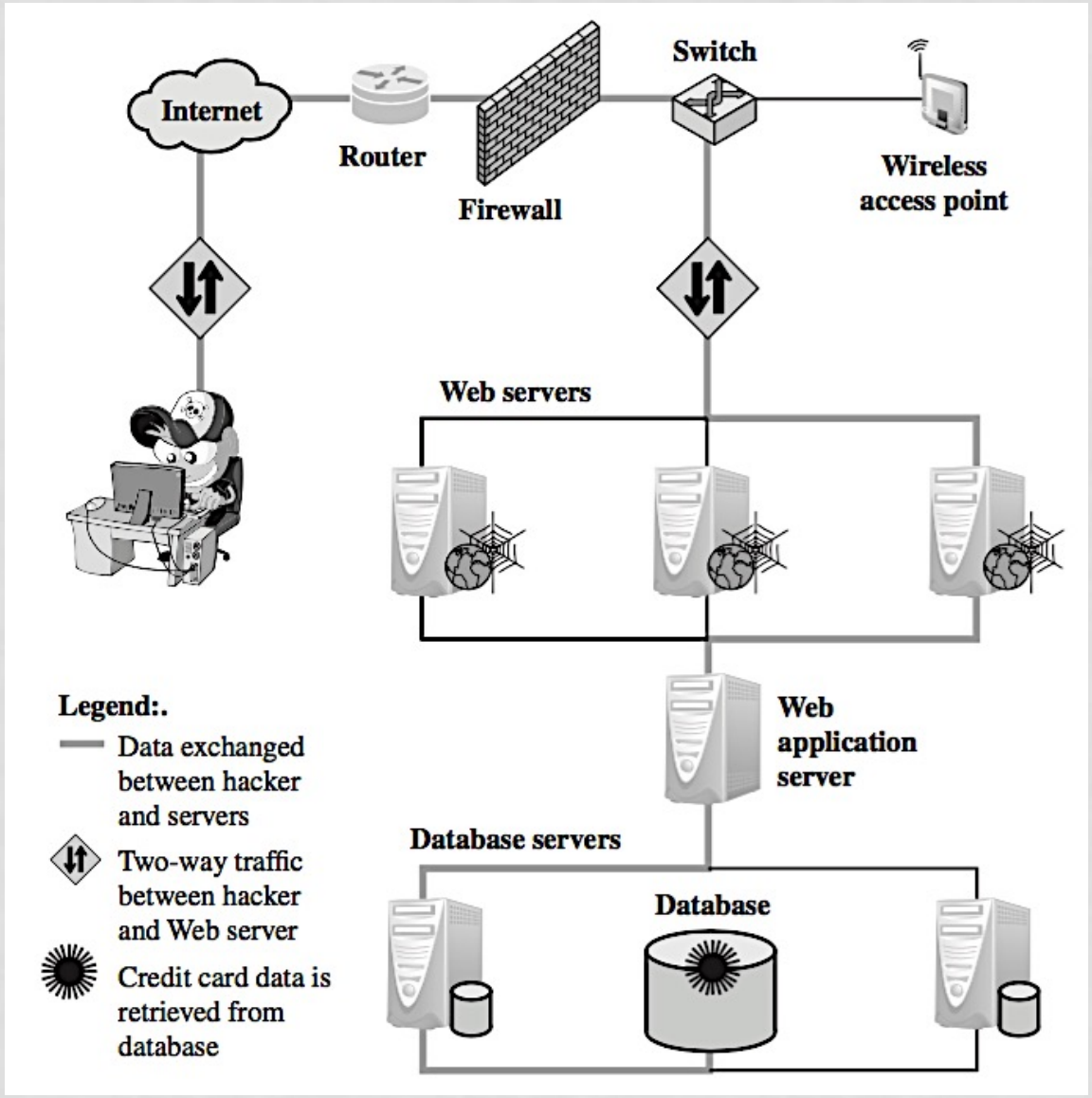
- https://www.net-security.org/secworld.php?id=13313
- "Many, many sites have lost customer data in this way," said Chris Hinkley, Senior Security Engineer at FireHost. "SQL Injection attacks are often automated and many website owners may be blissfully unaware that their data could actively be at risk. These attacks can be detected and businesses should be taking basic and blanket steps to block attempted SQL Injection, as well as the other types of attacks we frequently see."



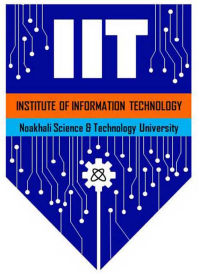
NEWS OF SQL ATTACKS

- http://www.mysqlperformanceblog.com/2012/07/18/sql-injection-still-a-problem/
- An SQL injection vulnerability resulted in an urgent June bugfix release of Ruby on Rails 3.x.
- Yahoo! Voices was hacked in July. The attack acquired 453,000 user email addresses and passwords. The perpetrators claimed to have used union-based SQL injection to break in.
- LinkedIn.com leaked 6.5 million user credentials in June. A class action lawsuit alleges that the attack was accomplished with SQL injection.
- SQL injection was documented as a security threat in 1998, but new incidents still occur every month. Making honest mistakes, developers fail to defend against this means of attack, and the security of online data is at risk for all of us because of it.

Typical SQL Injection Attack



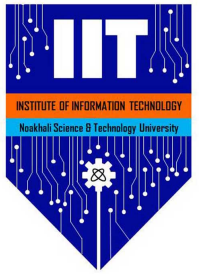
AN EXAMPLE SQL INJECTION ATTACK



Product Search: `blah' OR 'x' = 'x'`

- This input is put directly into the SQL statement within the Web application:
  - `$query = "SELECT prodinfo FROM prodtable WHERE prodname = '" . $_POST['prod_search'] . "'";`
- Creates the following SQL:
  - `SELECT prodinfo FROM prodtable WHERE prodname = 'blah' OR 'x' = 'x'`
  - Attacker has now successfully caused the entire database to be returned.

A MORE MALICIOUS EXAMPLE

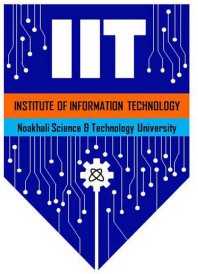


- What if the attacker had instead entered:
  - **`blah'; DROP TABLE prodinfo; --`**
- Results in the following SQL:
  - `SELECT prodinfo FROM prodtable WHERE prodname = 'blah'; DROP TABLE prodinfo; --'`
  - Note how comment (--) consumes the final quote
- Causes the entire database to be deleted
  - Depends on knowledge of table name
  - This is sometimes exposed to the user in debug code called during a database error
  - Use non-obvious table names, and never expose them to user
- Usually data destruction is not your worst fear, as there is low economic motivation

OTHER INJECTION POSSIBILITIES

- Using SQL injections, attackers can:
  - Add new data to the database
    - Could be embarrassing to find yourself selling politically incorrect items on an eCommerce site
    - Perform an INSERT in the injected SQL
  - Modify data currently in the database
    - Could be very costly to have an expensive item suddenly be deeply ‘discounted’
    - Perform an UPDATE in the injected SQL
  - Often can gain access to other user’s system capabilities by obtaining their password

SQLi ATTACK AVENUES AND TYPES



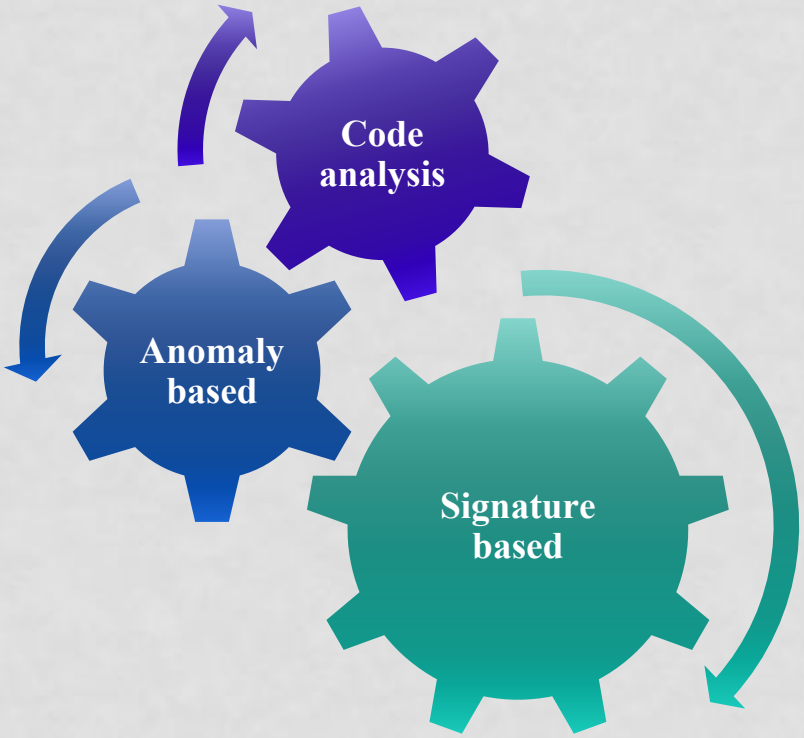
User Input	• attackers inject SQL commands by providing suitably crafted user input • user input typically comes from form submissions that are sent to the Web application via HTTP GET or POST requests
Server Variables	• If these variables are logged to a database without sanitization, this could create an SQL injection vulnerability • When the query to log the server variable is issued to the database, the attack in the forged header is then triggered
Second-order Injection	• A malicious user could rely on data already present in the system or database to trigger an SQL injection attack • Such attack does not come from the user, but from within the system itself
Cookies	• An attacker could alter cookies such that when the application server builds an SQL query based on the cookie's content, the structure and function of the query is modified
Physical User Input	• Such user-input could take the form of conventional barcodes, RFID tags, or even paper forms which are scanned using optical character recognition and passed to a database management system

SQLi COUNTERMEASURES

Defensive Coding

- Manual defensive coding practices
  - An example is input type checking, to check that inputs that are supposed to be numeric contain no characters other than digits
- Parameterized query insertion
- SQL DOM
  - query-building process

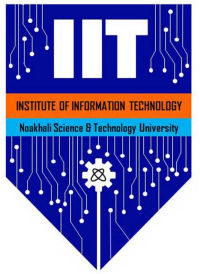
Detection



DATABASE ACCESS CONTROL

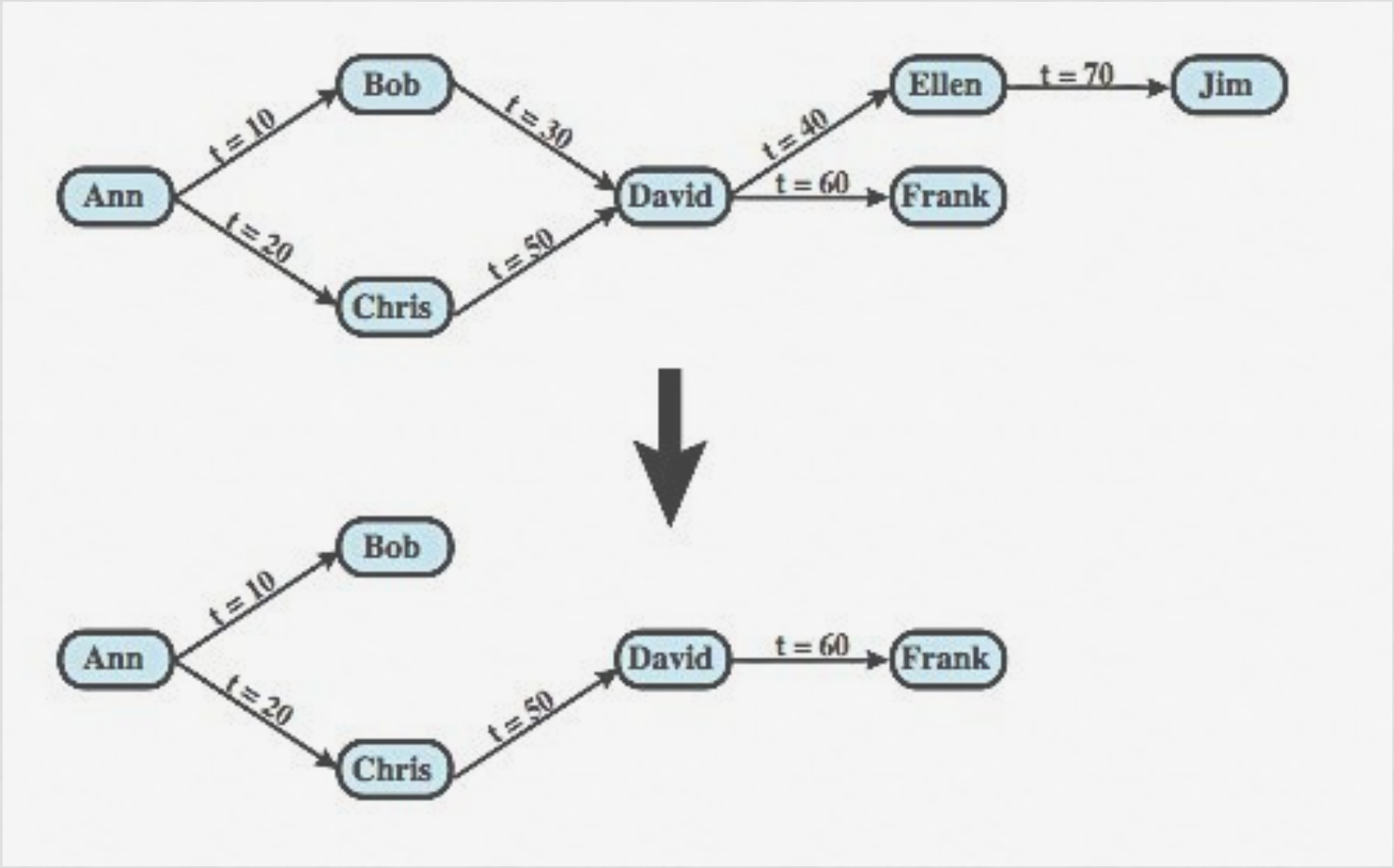
- DBMS provide access control for database
- Assume have authenticated user
- DBMS provides specific access rights to portions of the database
  - e.g. Create, Insert, Delete, Update, Read, Write
  - To entire database, tables, selected rows or columns
  - Possibly dependent on contents of a table entry
- Can support a range of policies:
  - Centralized administration
  - Ownership-based administration
  - Decentralized administration

SQL ACCESS CONTROLS



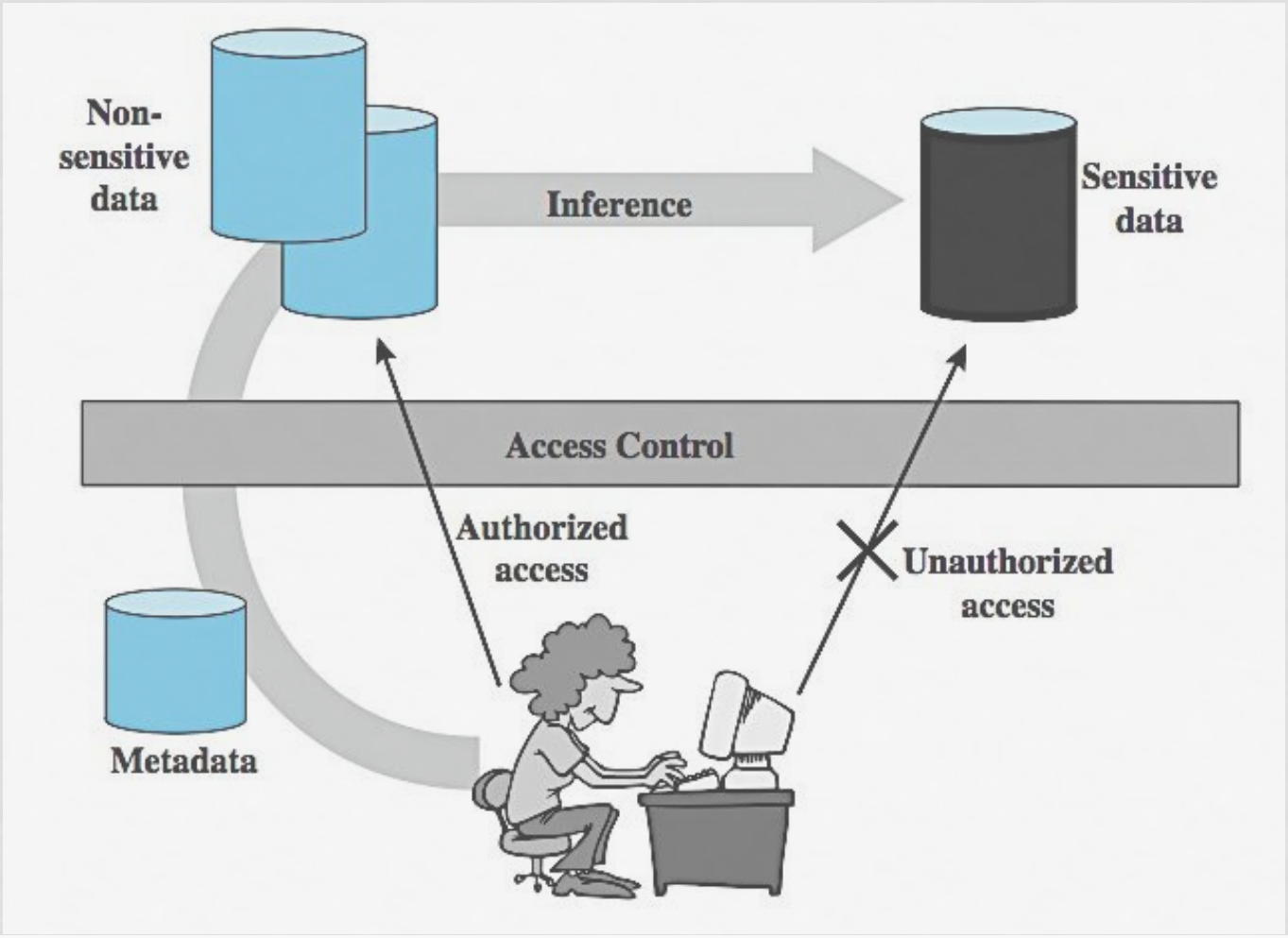
- two commands:
  - `GRANT { privileges | role } [ON table] TO { user | role | PUBLIC } [IDENTIFIED BY password] [WITH GRANT OPTION]`
    - e.g. GRANT SELECT ON ANY TABLE TO ricflair
  - `REVOKE { privileges | role } [ON table] FROM { user | role | PUBLIC }`
    - e.g. REVOKE SELECT ON ANY TABLE FROM ricflair
- typical access rights are:
  - `SELECT, INSERT, UPDATE, DELETE, REFERENCES`

CASCADING AUTHORIZATIONS



ROLE-BASED ACCESS CONTROL

- Role-based access control work well for DBMS
  - Eases admin burden, improves security
- Categories of database users:
  - Application owner
  - End user
  - Administrator
- DB RBAC must manage roles and their users
  - Cf. RBAC on Microsoft's SQL server



INFERENCE

Indirect Information Access via Inference Channel

INFERENCE EXAMPLE

Name	Position	Salary ($)	Department	Dept. Manager
Andy	senior	43,000	strip	Cathy
Calvin	junior	35,000	strip	Cathy
Cathy	senior	48,000	strip	Cathy
Dennis	junior	38,000	panel	Herman
Herman	senior	55,000	panel	Herman
Ziggy	senior	67,000	panel	Herman

(a) Employee table

```
CREATE view V1 AS
SELECT Availability,
       Cost
FROM Inventory
WHERE Department =
"hardware"
```

Position	Salary ($)
senior	43,000
junior	35,000
senior	48,000

Name	Department
Andy	strip
Calvin	strip
Cathy	strip

```
CREATE view V2 AS
SELECT Item,
       Department
FROM Inventory
WHERE Department =
"hardware"
```

(b) Two views

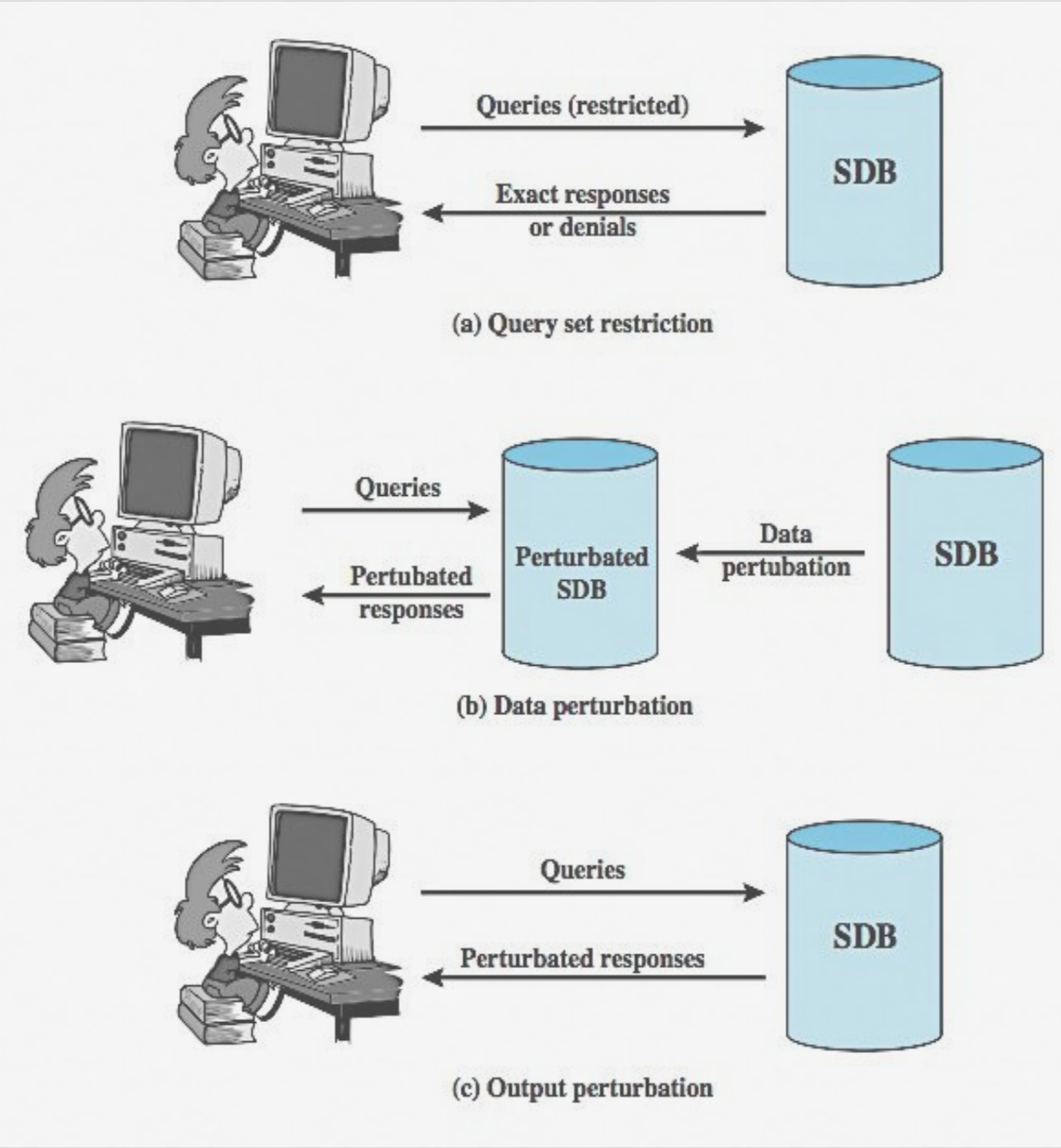
Name	Position	Salary ($)	Department
Andy	senior	43,000	strip
Calvin	junior	35,000	strip
Cathy	senior	48,000	strip

(c) Table derived from combining query answers

INFERENCE COUNTERMEASURES

- Inference detection at database design
  - Alter database structure or access controls
- Inference detection at query time
  - By monitoring and altering or rejecting queries
- Need some inference detection algorithm
  - A difficult problem
  - Cf. Employee-salary example

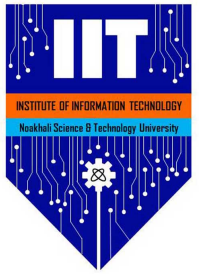
PROTECTING AGAINST INFERENCE



STATISTICAL DATABASES

- Provides data of a statistical nature
  - e.g. Counts, averages
- Two types:
  - Pure statistical database
  - Ordinary database with statistical access
    - Some users have normal access, others statistical
- Access control objective to allow statistical use without revealing individual entries
- Security problem is one of inference

STATISTICAL DATABASE SECURITY



- Use a characteristic formula C
  - A logical formula over the values of attributes
  - e.g. `(gender=male) AND ((major=cs) OR (major=ee))`
- Query set X(*C*) of characteristic formula *C*, is the set of records matching C
- A statistical query is a query that produces a value calculated over a query set

STATISTICAL DATABASE EXAMPLE

(a) Database with statistical access with $N = 13$ students

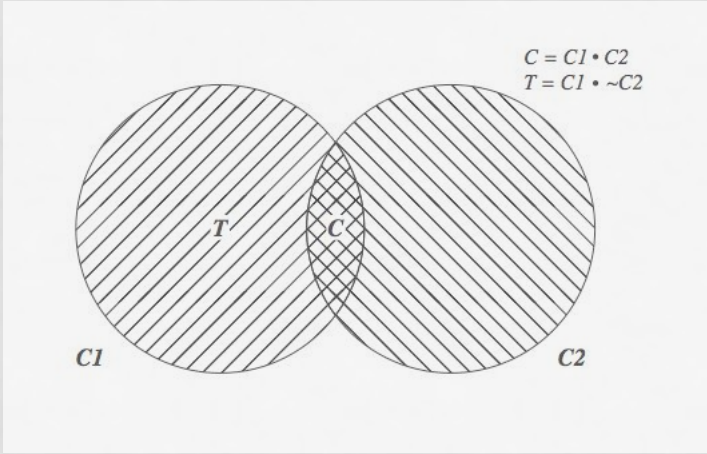
Name	Sex	Major	Class	SAT	GP
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	Psy	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	Psy	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5

(b) Attribute values and counts

Attribute A_j	Possible Values	$\lvert A_j \rvert$
Sex	Male, Female	2
Major	Bio, CS, EE, Psy, ...	50
Class	1978, 1979, 1980, 1981	4
SAT	310, 320, 330, ..., 790, 800	50
GP	0.0, 0.1, 0.2, ..., 3.9, 4.0	41

TRACKER ATTACKS

- Divide queries into parts
  - C = C1.C2
  - Count(C.D) = count(C1) - count (C1. ~*C2)*
- Combination is called a tracker
- Each part acceptable query size
- Overlap is desired result



OTHER QUERY RESTRICTIONS

- Query Set Overlap Control
  - Limit Overlap Between New & Previous Queries
  - Has Problems And Overheads
- Partitioning
  - Cluster Records Into Exclusive Groups
  - Only Allow Queries On Entire Groups
- Query Denial And Information Leakage
  - Denials Can Leak Information
  - To Counter Must Track Queries From User

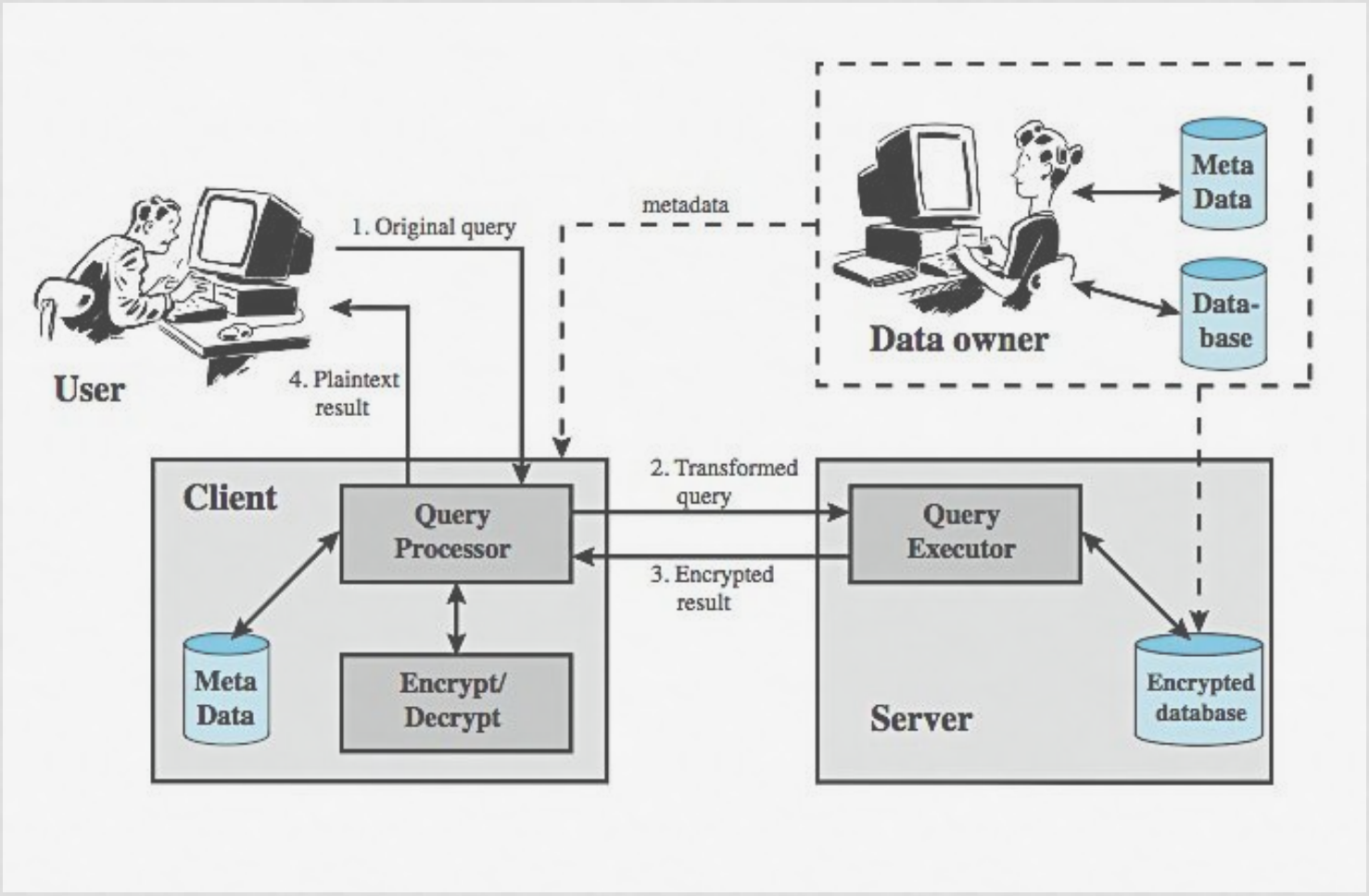
PERTURBATION

- Add Noise To Statistics Generated From Data
  - Will Result In Differences In Statistics
- Data Perturbation Techniques
  - Data Swapping
  - Generate Statistics From Probability Distribution
- Output Perturbation Techniques
  - Random-sample Query
  - Statistic Adjustment
- Must Minimize Loss Of Accuracy In Results

DATABASE ENCRYPTION

- Databases Typical A Valuable Info Resource
  - Protected By Multiple Layers Of Security: Firewalls, Authentication, O/S Access Control Systems, DB Access Control Systems, And Database Encryption
- Can Encrypt
  - Entire Database - Very Inflexible And Inefficient
  - Individual Fields - Simple But Inflexible
  - Records (Rows) Or Columns (Attributes) - Best
    - Also Need Attribute Indexes To Help Data Retrieval
- Varying Trade-offs

DATABASE ENCRYPTION



ENCRYPTED DATABASE EXAMPLE

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

For example, the ***eid*** values could be partitioned by mapping [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000] into 2, 3, 5, 1, and 4, respectively

E(k, B)	I(eid)	I(ename)	I(salary)	I(addr)	I(did)
1100110011001011…	1	10	3	7	4
0111000111001010…	5	7	2	7	8
1100010010001101…	2	5	1	9	5
0011010011111101…	5	5	2	4	9

SUMMARY

- Introduced databases and DBMS
- Relational databases
- SQL injection
- Database access control issues
  - SQL, role-based
- Inference
- Statistical database security issues
- Database encryption