Systems Design and Security

UNIFIED MODELING LANGUAGE

Part 10: Security and Robustness

http://staffwww.dcs.shef.ac.uk/people/A.Simons/

Home ⇒ Teaching ⇒ Lectures ⇒ COM2008/COM3008

The University Of Sheffield.

Bibliography

- Software Engineering
  - I Sommerville. Software Engineering, 10th ed., Pearson, 2022.
- Database Systems
  - T Connolly and C Begg, Database Systems – a Practical Approach to Design, Implementation and Management, 6th ed., Pearson, 2014.
- Security Engineering
  - R Anderson. Security Engineering, 3rd ed., John Wiley, 2022. https://www.cl.cam.ac.uk/~rja14/book.html

Outline

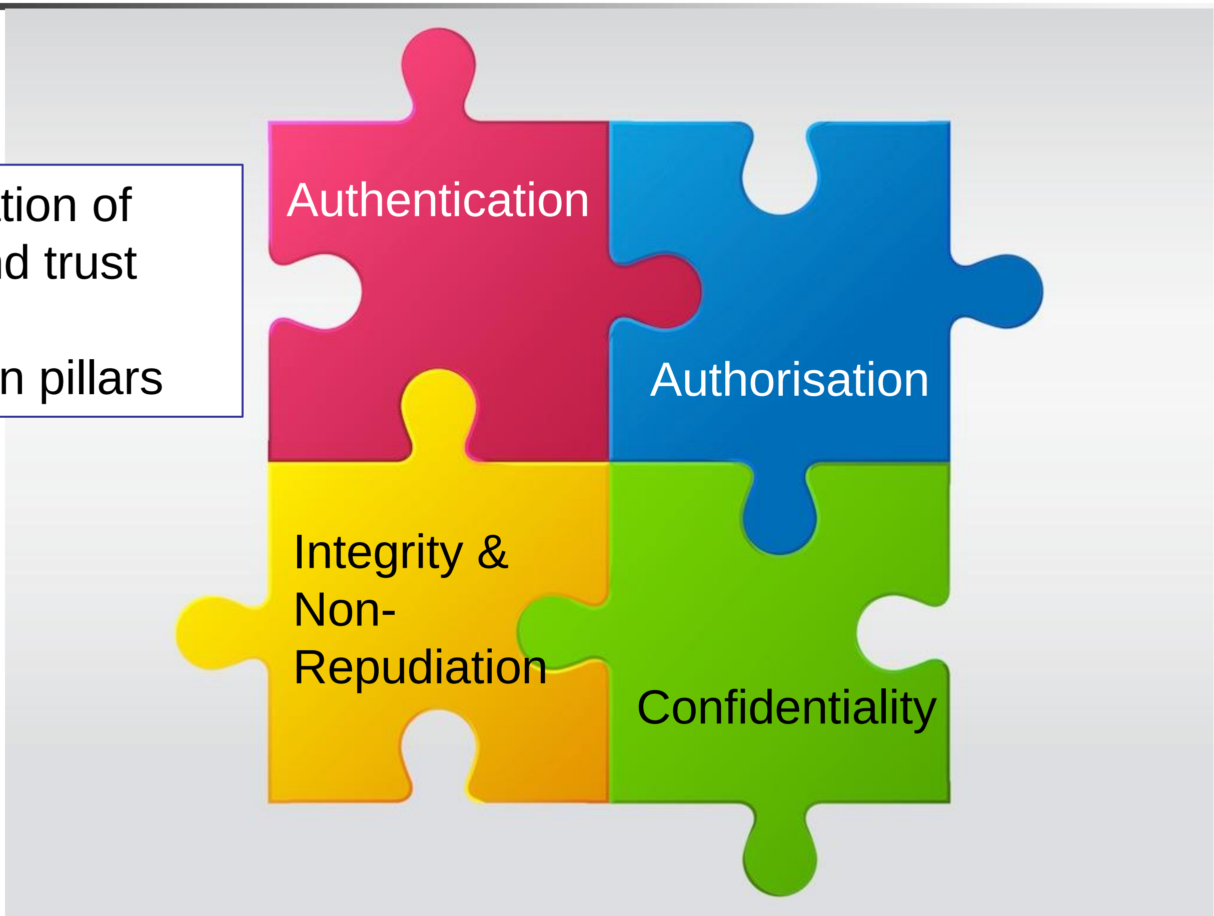
- Authentication and authorisation
- Integrity and non-repudiation
- Confidentiality and encryption
- Robustness via redundancy
- Restricted access to views
- Penetration resistance
- Availability via concurrency

Reading: Sommerville chapters 10-14;
Connolly and Begg, chapters 20, 22, 24-26;
Anderson – the whole book!

What is Security?

Combination of safety and trust

Four main pillars



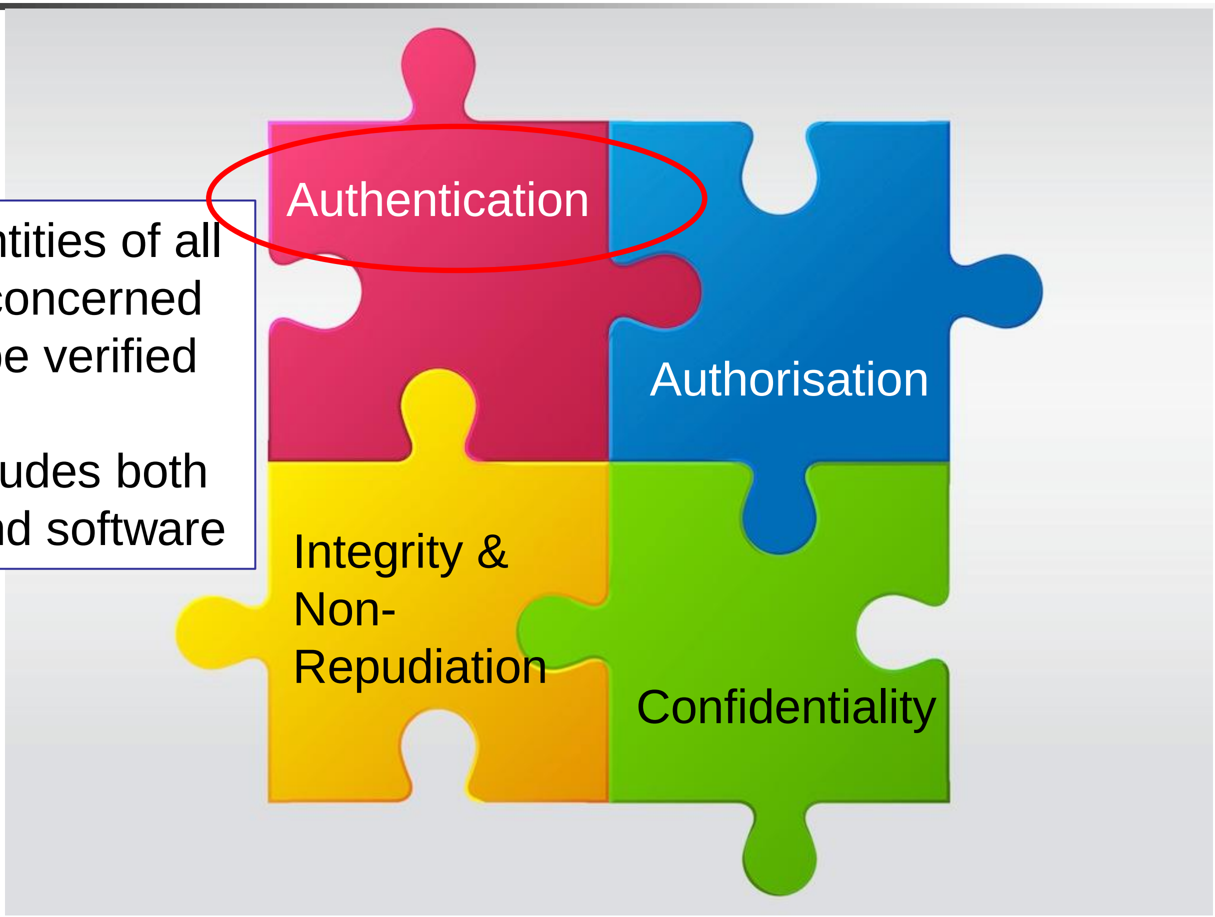
http://www.freevector.com/jigsaw-puzzle

The University Of Sheffield.

Authentication

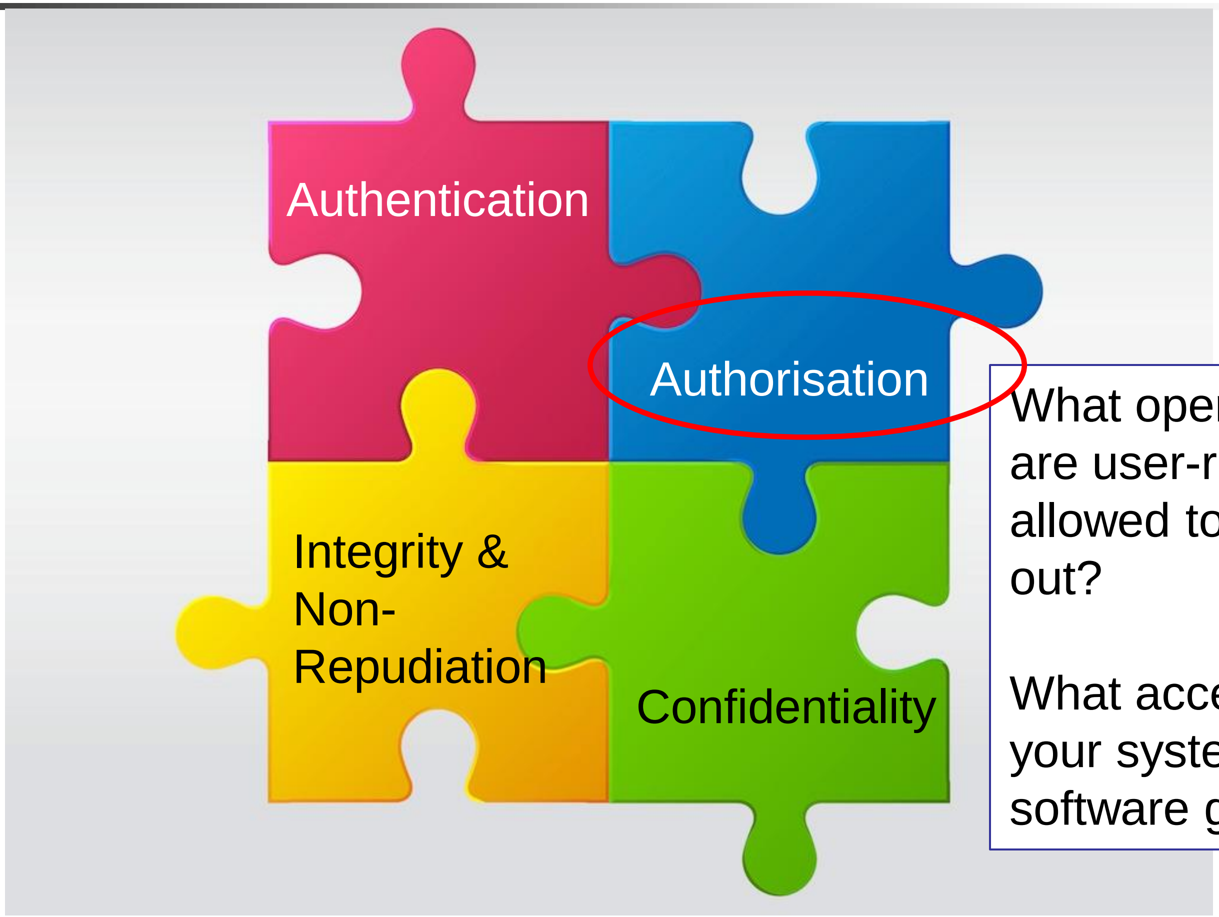
The identities of all parties concerned should be verified

This includes both users and software



http://www.freevector.com/jigsaw-puzzle

Authorisation



http://www.freevector.com/jigsaw-puzzle

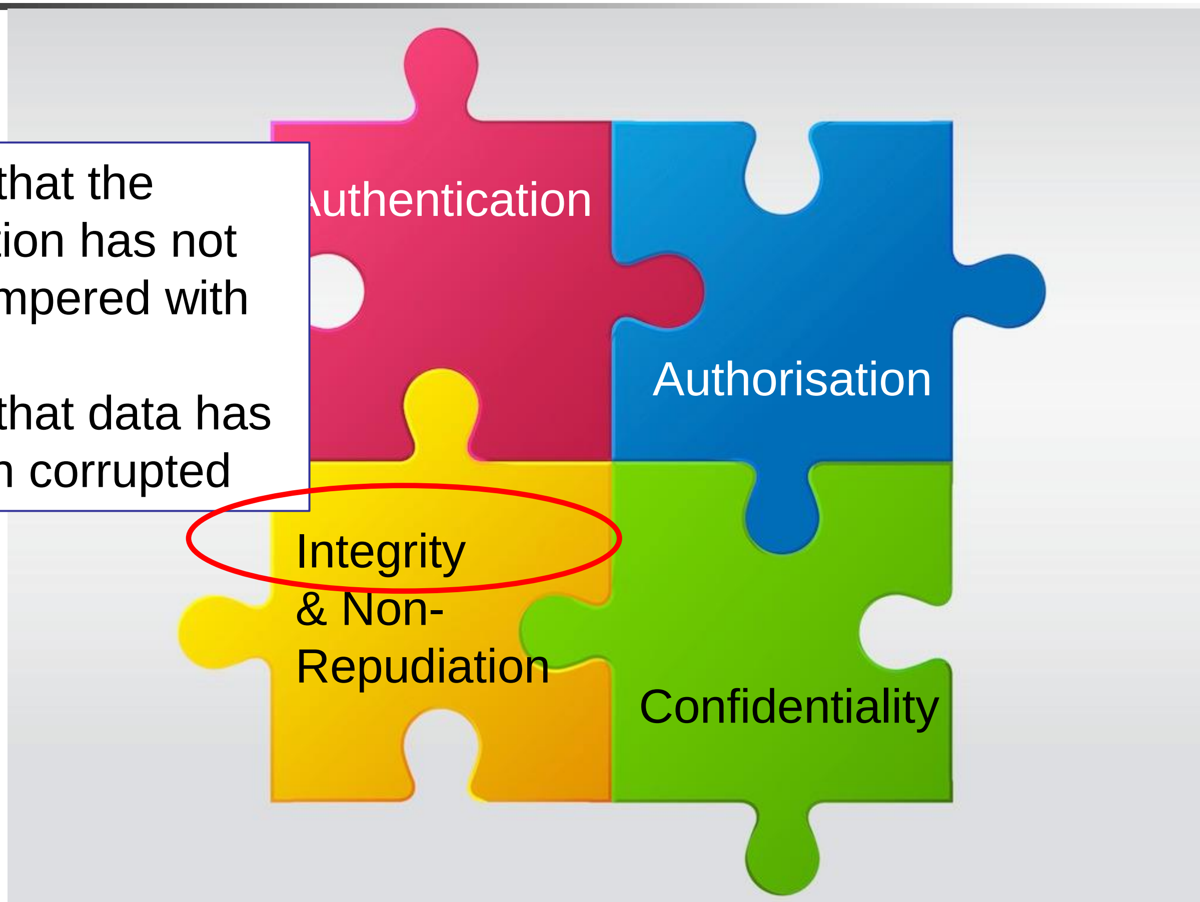
What operations are user-roles allowed to carry out?

What access to your system is the software granted?

Integrity

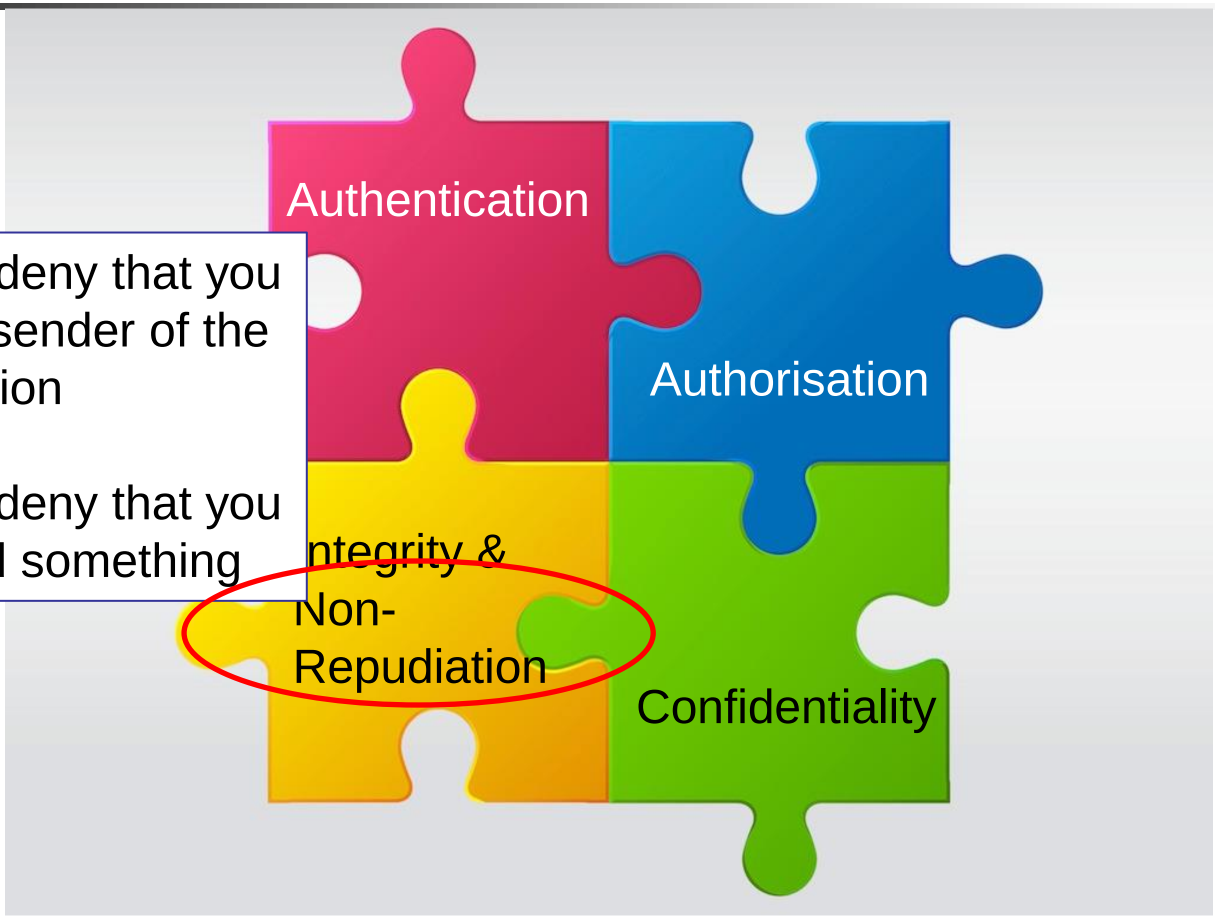
Ensure that the information has not been tampered with

Ensure that data has not been corrupted



http://www.freevector.com/jigsaw-puzzle

Non-Repudiation

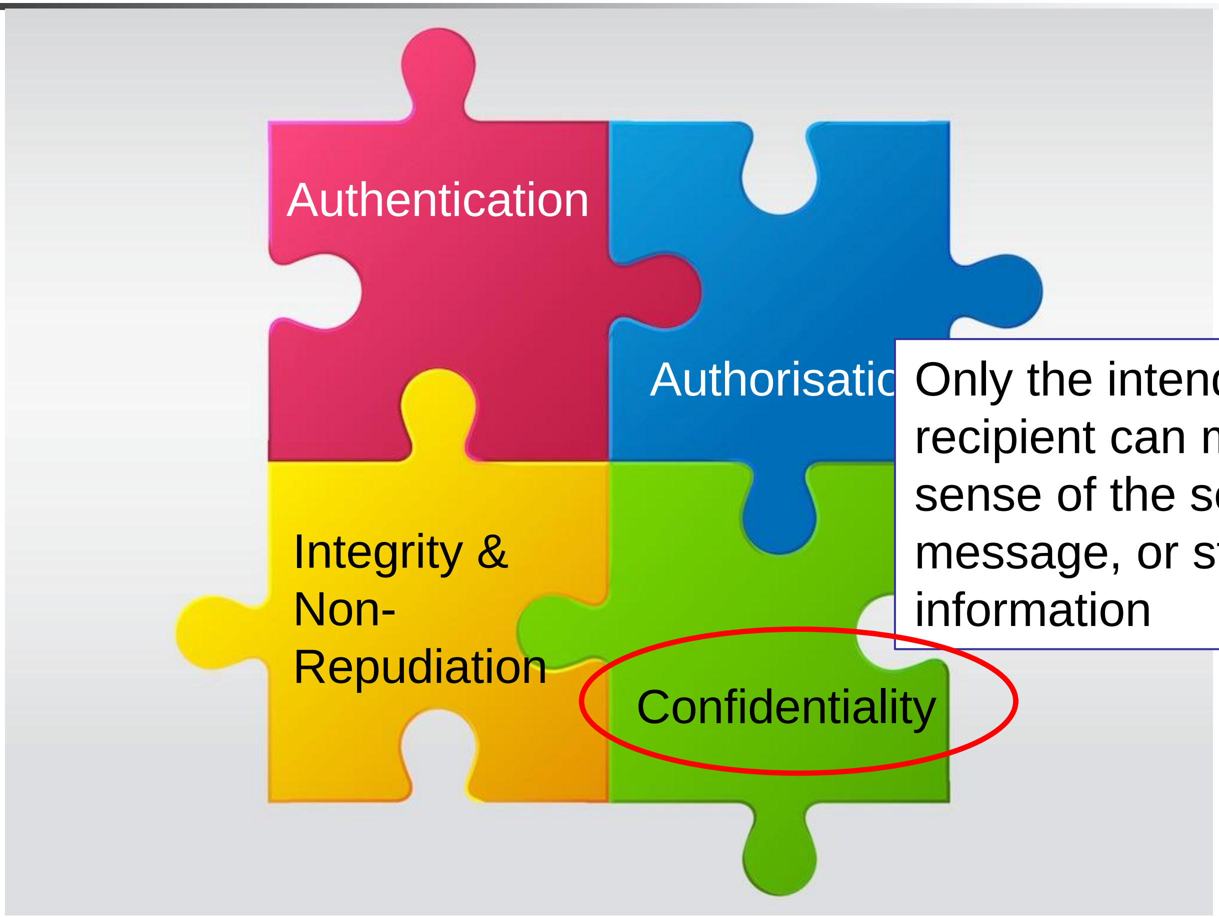


Cannot deny that you are the sender of the information

Cannot deny that you received something

http://www.freevector.com/jigsaw-puzzle

Confidentiality



http://www.freevector.com/jigsaw-puzzle

Only the intended recipient can make sense of the sent message, or stored information

Tradeoffs

- With unlimited resources, most forms of security can eventually be broken
  - arms race between the criminal and the security expert
- Cost of breaking must outweigh the reward
  - make it harder for them to crack your system (compared to others)
- Must consider end-to-end security
  - only as secure as the weakest link
  - eg: you have private key encryption, but how secure is your key store?
- Aim for simplicity
  - you don't want to frighten away legitimate users
  - security regime must be easy to maintain

Security Scenarios

- Online banking
  - authentication: is this a valid account holder?
  - authorisation: do they have permission to access this account?
  - confidentiality: is the account data secure?
  - is security vetting easy for legitimate users?
- Downloading software
  - authentication: does the code come from a trusted source?
  - integrity: has the code been tampered with, before or during downloading?
  - authorisation: does the code have permission to install itself on your machine, or remove other files?
  - will the software only do what it claims to do?

Security Scenarios

- Online credit transactions
  - authentication:
    - does the credit card belong to the customer?
    - is the merchant valid?
    - is the merchant bank valid?
  - integrity:
    - have any details been altered *en route*?
  - non-repudiation:
    - can the purchaser deny that they ordered the item?
    - can the vendor deny that they received the money?
  - confidentiality:
    - should the merchant have access to your credit card?
    - should the bank have access to the purchase details?

Authentication

- Password-controlled access
  - every user given a unique loginID
  - every user given a secret password
  - password of 8+ chars, mixed case, digits and symbols
- Pre-registration schemes
  - e.g. HTTP htaccess: 64-bit encrypted user:password
  - e.g. HTTP digest: better encryption of user-info
  - server decodes against a fixed list of known users
- Self-registration schemes
  - form-based self-registration, dynamic list of known users
  - requires secure transmission, storage of passwords
  - e.g. encrypted with additional "salt" data in DB
  - requires super-user determination of role-based access (next)

Authorisation

- Mandatory Access Control (MAC)
  - controlled by a security manager in high-security systems
  - e.g. permission for a thread to touch certain data
- Discretionary Access Control (DAC)
  - permissions set as desired by the data owner
  - e.g. Unix user, group and world access rights to files
- Role-based Access Control (RBAC)
  - certain users are granted certain roles
  - each role is granted certain permissions
  - these concern different target objects (files, ports)
- Lattice-based Access Control (LBAC)
  - a user must exceed access level of protected accessible object
  - multiple users granted greatest lower bound (meet) access
  - multiple objects offer least upper bound (join) access

Role-Based

- Who or what?
  - individual users, or groups
  - role-based access, eg: manager, customer
  - specific programs, from a given origin
- Which resource?
  - individual files, DB tables
  - files of certain template-types
  - files from certain locations
- What restriction?
  - read only
  - read and write
  - update or remove files
  - connect, listen to channel

- Bespoke methods grant role-based permissions to users
- Java also has features to support authorisation for software

Restricted Users

- SQL privileges grant selective access to different users

```
GRANT SELECT ON MyDB.Borrower
  TO PUBLIC;
```

allows anyone to select on the Borrower table in MyDB

```
CREATE USER 'user1'@'localhost'
  IDENTIFIED BY 'passwd1';
```

creates user1 with passwd1

```
GRANT ALL ON MyDB.Loan
  TO 'user1'@'localhost';
```

allows user1 to insert, delete, select, update the Loan table

```
GRANT SELECT, UPDATE(issueDate, dueDate)
  ON MyDB.Loan TO 'user1'@'localhost';
```

allows user1 to select, update two columns of the Loan table

```
GRANT UPDATE ON MyDB.BookTitle TO
  'user1'@'localhost' WITH GRANT OPTION;
```

allows user1 to update BookTitle; and to pass on this privilege to others

Restricted Roles

- SQL roles grant groups of privileges to groups of users

CREATE ROLE 'SuperUser', 'User';	defines two roles named SuperUser and User
GRANT ALL ON MyDB.* TO 'SuperUser';	allows SuperUser role to have all permissions on MyDB
GRANT SELECT ON MyDB.* to 'User';	allows User role only to have viewing permissions
GRANT 'SuperUser' TO 'user1'@'localhost'; GRANT 'User' TO 'user2'@'localhost'; GRANT 'User' TO 'user3'@'localhost';	grants roles with the given permissions to individual users
REVOKE 'User' FROM 'user2'@'localhost';	revoke roles and permissions

Lab 1: User Accounts

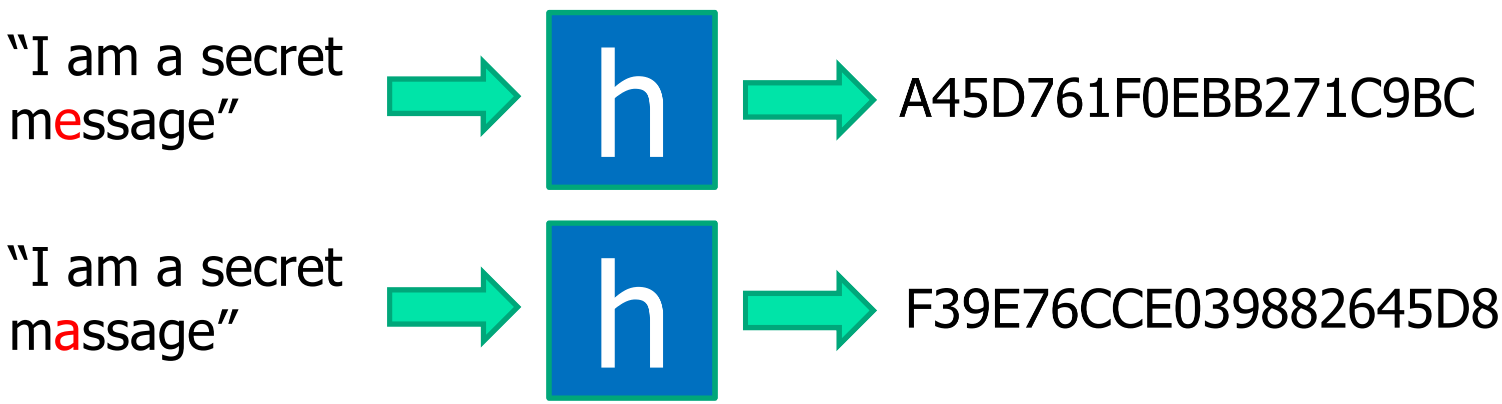
Run a Poll

- How will your database set up user accounts?
  - your MySQL DB has only one secure login for the team, with a team password (secret in your Java code)
  - you cannot create extra MySQL accounts on stustore
  - does your application support self-registration?
  - how will you handle setting up new user accounts?
  - how will new users be assigned suitable privileges?
- What tables or Java code will you need:
  - to handle usernames and passwords?
  - to offer role-based access permissions?

Integrity

- Secure transmission
  - authentication: comes from a valid source
  - integrity: has not been modified/corrupted in transit
- Main technologies
  - authentication: digital signatures, certificates
  - integrity: message digests (digital fingerprints)
- Both use encryption
  - Secure Hash Algorithm (SHA) for one-way encoding of message digests - evidence of no tampering
  - shared key encryption (but problem of key sharing, storage)
  - Rivest-Shamir-Adleman (RSA) algorithm for public/private key encryption - avoids problem of key storage and sharing

Cryptographic Hash



- keyless algorithm, easy to compute
- fixed-length digests, 100-200 bits
- hard to recover the clear message
- hard to obtain a collision, h(m1) = h(m2)
- discontinuous mapping, h(m) <> h(m′)

Secure Hash Algorithm

- SHA-1 from NSA and NIST
  - block size 512 bits
- generates 160-bit hash
  - 2^{160} digests
- previous competitors
  - MD4 (weak)
  - MD5 128-bit hash
- how secure?
  - attack found in 2005
  - from 2010, deprecated
  - SHA-2, SHA-3 now recommended

```
int a, b, c, d, e = ...

for (t=0; t < 79; ++t) {
  int tmp = c << 5 +
    f(b,c,d) + e + W[t]
    + K[t];
  e = d;
  c =  b << 30;
  b = a;
  a = tmp;
}

// arithmetic is mod 32
```

Message Authentication

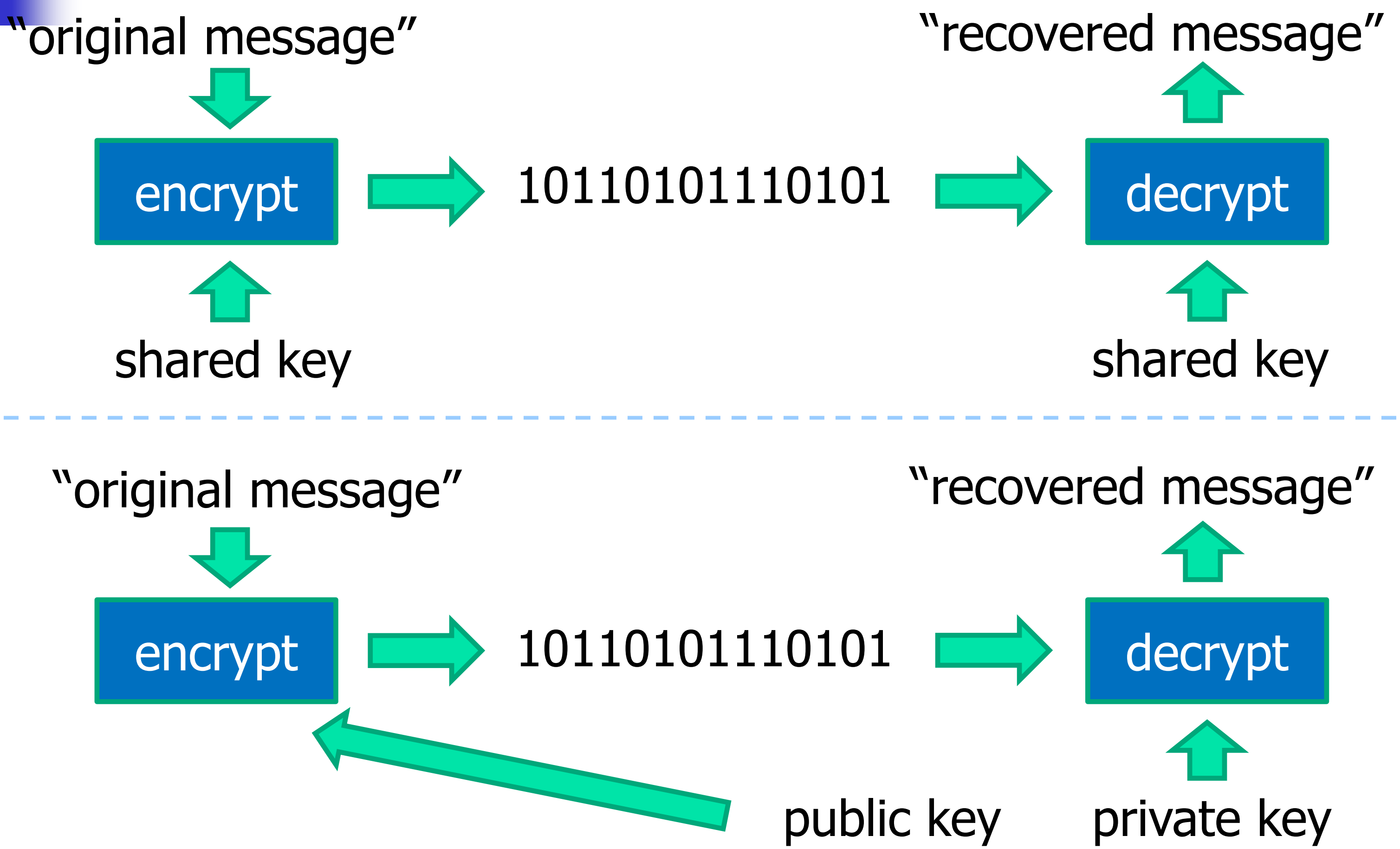
- Simple message digest
- sender sends:
  - message m
  - digest d = h(m)
- receiver computes
  - digest d’ = h(m)
  - checks d = d’
- weakness
  - if both d, m modified in transit

- MAC: Message authentication code
  - shared password p
- sender computes:
  - digest d = h(p || m)
  - mac = h(p || d)
  - sends m, mac
- receiver computes
  - digest d’ = h(p || m)
  - checks mac = h(p || d’)

Confidentiality

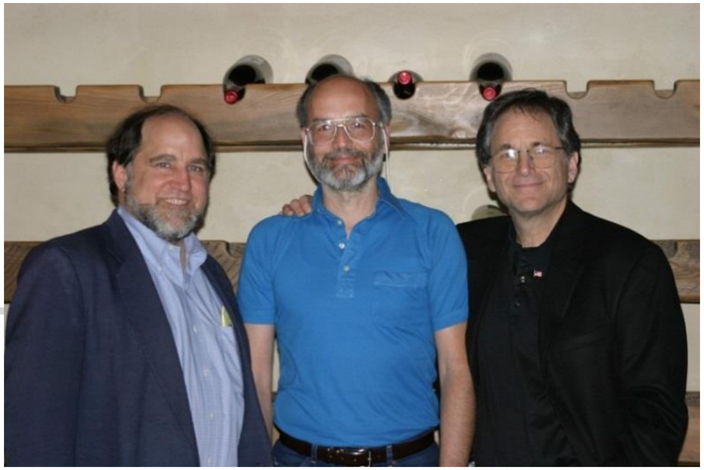
- Symmetric key encryption
  - use same key to encrypt and decrypt
  - or determine one key from the other
  - problem with secure key sharing, storage
  - needs new key for each pair of parties
  - key management problem
- Public/private key encryption
  - uses two related prime keys, very large
  - impossible to guess one from the other
  - sender encrypts with receiver's public key
  - receiver decrypts with own private key
  - no key transmission issues

Encryption / Decryption



RSA Algorithm

Ron Rivest, Adi Shamir, Len Adleman

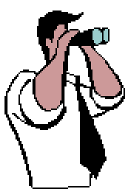


- Choose n, the product of two large primes p and q
- Choose e and d with very specific mathematical properties in relation to n
  - e.g. gcd(d, p-1)=1, gcd(e, q-1)=1
- e and n form the public key, released
- d is the private key, never released
- message m, is encrypted to cipher c, as
  - $c = m^e \bmod n$
- cypher c is decrypted, yielding message m, as
  - $m = c^d \bmod n$

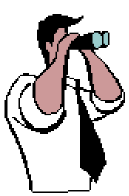
http://www.usc.edu/dept/molecular-science/RSA-2003.htm

Cryptanalysis

- 1974 Data Encryption Standard (IBM)
  - 56-bit key cracked in 22hrs in1999 using 100,000 computers; decertified; is probably easily cracked now
  - 64-bit keys breakable by governments, criminal organisations
  - 128-bit keys good for foreseeable future (unless quantum comp)



- 1976 Diffie & Hellmann public key cryptograpy
  - prime factors of large products (> 10^{100}) very hard to find, but need large keys > 512 bits
  - Sarah Flannery's algorithm speeds up processing x30 in 1999



- 1977 RSA (Rivest-Shamir-Adleman) public/private key algorithm
  - actually GCHQ spy Clifford Cocks had this from 1973, secret till 1997!
  - 512-bit keys can be cracked in weeks using dedicated machines
  - 1024-bit, 2048-bit keys good for many decades?

Signatures, Certificates

- Digital Signature
  - combination of message digest and encryption
  - sender encrypts digest with own private key
  - receiver decrypts digest with sender's public key
  - authenticates the sender, and message integrity
- Digital Certificate
  - certifying authority (CA) vouches for third party by issuing certificate with public key
  - first party trusts CA, so uses this public key to authenticate third party
  - example CAs: Verisign, Entrust

Java Security

- Java Sandbox
  - a "safe environment" in which to run Java code
  - can prevent reading/writing to files on host machine
  - originally for applets; now for any Java application
  - uses a security manager (disabled by default)
- Security elements
  - permissions: actions that code is allowed to perform
  - code sources: origin of code, digital signatures
  - protection domains: dictionary that maps permissions to code sources
  - key stores: digital signatures use keys, held in store
  - policy files: list permissions, identify keystore location, specify code sources and protection domains

Java Policy File

```
keystore "${user.home}${/}.keystore";

grant codeBase "http://www.dcs.sheffield.ac.uk/" {
  permission java.io.FilePermission "/tmp", "read";
  permission java.lang.RuntimePermission "queuePrintJob";
};
grant signedBy "ajhs" codeBase "http://ajhs.com/" {
  permission java.security.AllPermission;
};
grant signedBy "gjb" {
  permission java.net.SocketPermission "*:1024-",
    "accept, connect, listen";
};
grant {
  permission java.util.PropertyPermission
    "java.version", "read";
};
```

code from a given origin

a type of permission

Protection Domains

```
keystore "${user.home}${/}.keystore

grant codeBase "http://www.dcs.sheffield.ac.uk/" {
  permission java.io.FilePermission "/tmp", "read";
  permission java.lang.RuntimePermission "queuePrintJob";
};
grant signedBy "ajhs" codeBase "http://ajhs.com/" {
  permission java.security.AllPermission;
};
grant signedBy "gjb" {
  permission java.net.SocketPermissio
    "accept, connect, listen";
};
grant {
  permission java.util.PropertyPermi
    "java.version", "read";
};
```

allow code loaded from DCS to read /tmp and to queue print jobs

allow code loaded from ajhs.com and signed by ajhs to do anything

allow code signed by gjb to do 3 actions on all hosts, ports > 1024

allow all code to read the Java version

Java Example

```java
import java.net.*;
import java.security.*;

class AccessTest {
  public static void main(String[] args) {
    SocketPermission sp = new SocketPermission(
      "my.host.name:6000", "connect");
    try {
      AccessController.checkPermission(sp);
      System.out.println("OK to open socket");
      // carry on here ...

    }
    catch (AccessControlException ace) {
      ace.printStackTrace();
    }
  }
}
```

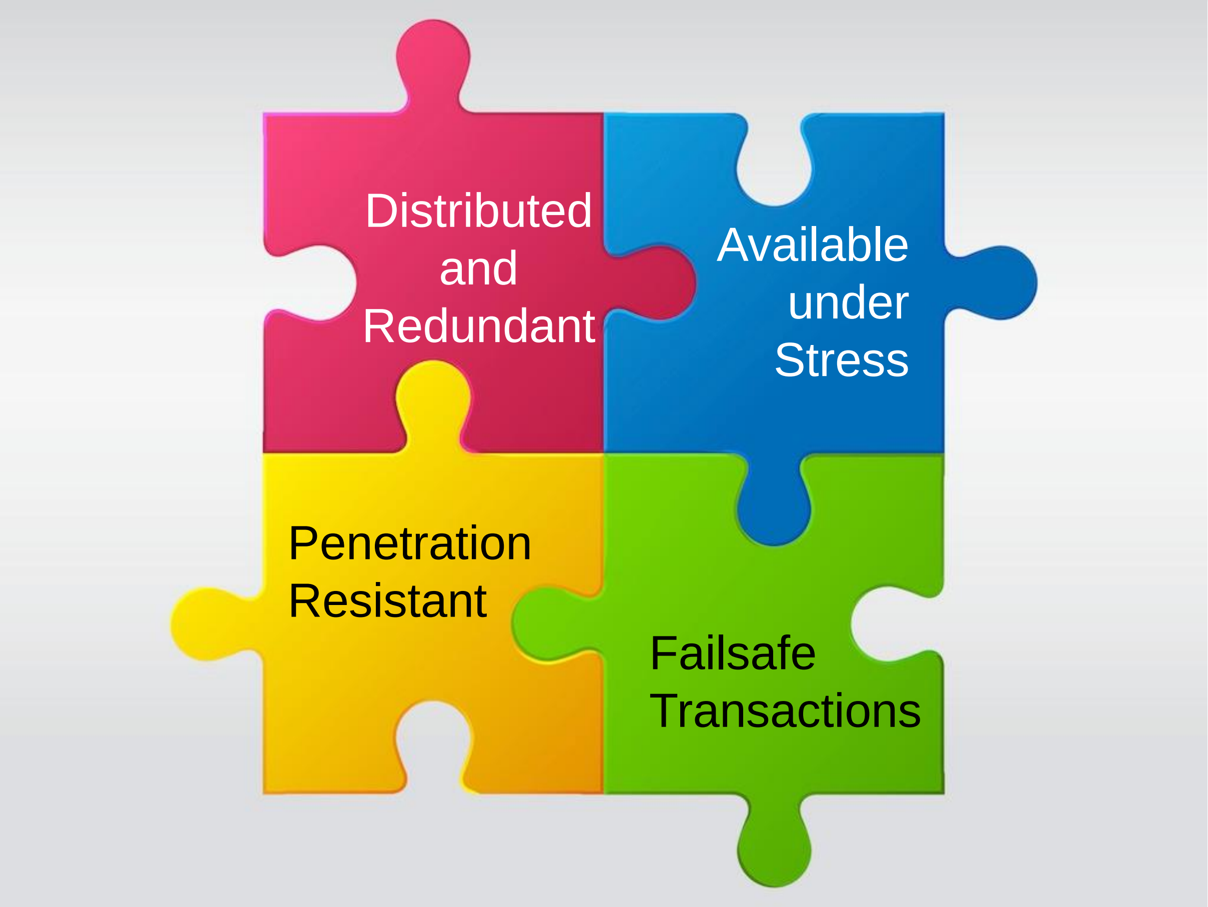
create a permission

test the permission

exit if refused

extend java.security.BasicPermission to create your own kinds of permission

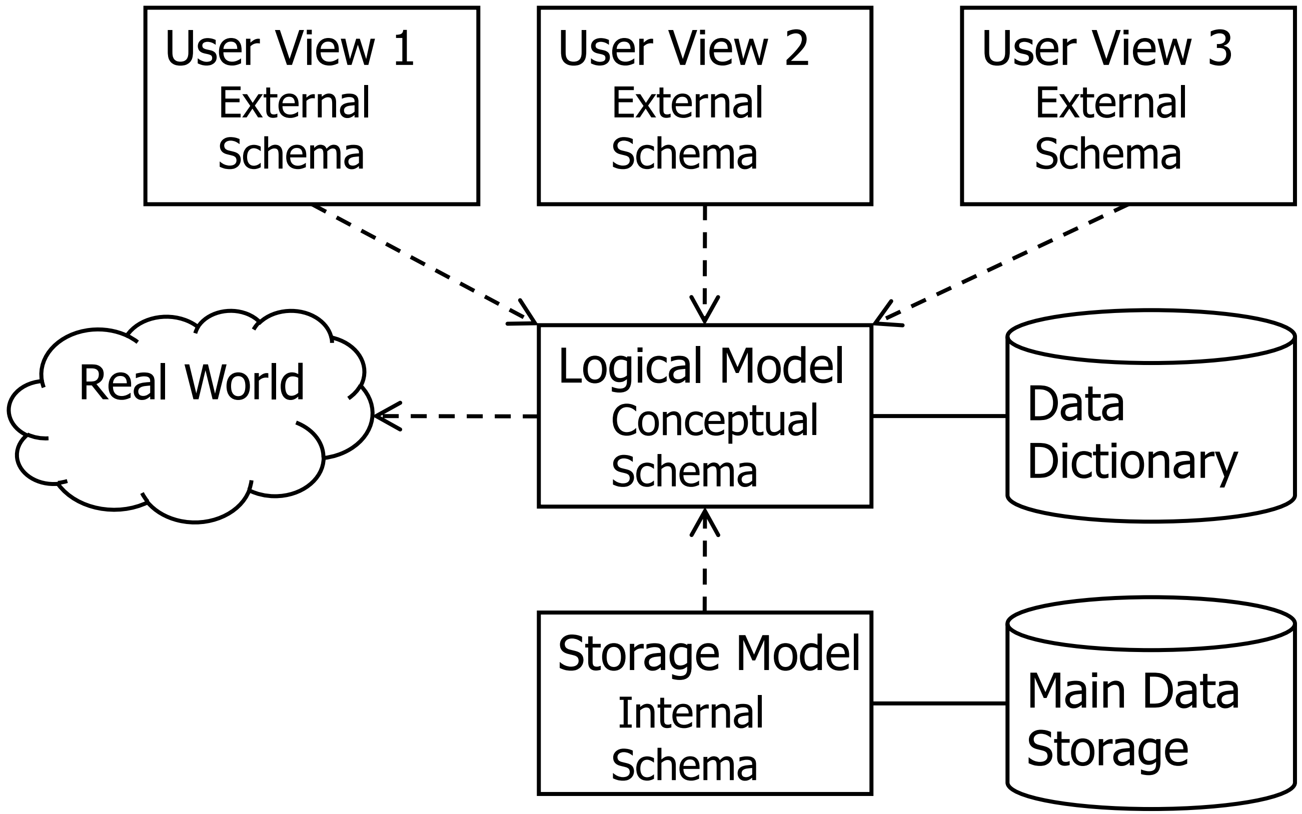
What is Robustness?



Physical Protection

- Protect against loss of main data centre
  - multiple copies of database in separate locations
    - but expensive to update live data simultaneously
  - regular backups of whole dataset to remote locations
  - data storage in "the cloud" – no more big office servers?
    - trust issues – who else can see data in the cloud?
- Protect against local hardware failure
  - Use RAID architecture (Redundant Array of Independent Disks)
  - RAID-0: data is "striped" (segmented) across different disks
  - RAID-1: mirror copy of data is also made, across different disks
  - RAID-2: error-correcting codes also retained, on different disks

Controlled Views



View Definition

- A view is a derived table, computed from other tables
- Views enhance usability, security in large, complex databases
- Can be inefficient to query – hides complex subquery

```
CREATE VIEW Overdue (memberID, forename, surname, overdue) AS
  SELECT memberID, forename, surname, COUNT (*) AS overdue
    FROM Borrower, Loan WHERE dueDate < '2022-10-01'
      GROUP BY memberID, forename, surname
```

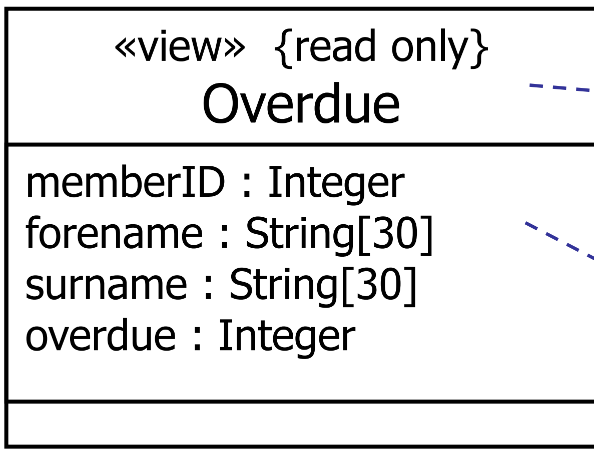
must name all non-aggregate columns in the grouping clause

Overdue

memberID	forename	surname	overdue
1012234	John	Smith	4
1667753	Jane	Doe	1
1784532	Jaswinder	Singh	3

The University Of Sheffield.

UML Profile for View



views are only updatable if they have a single base table and include the same PK

- UML Profiles
  - UML database profile is not universal; but widespread
- View tables
  - use the «view» label to indicate a view table, and
  - {read only} constraint if the view cannot be updated
- Any key attributes?
  - only if using same PK as the single base table
  - candidate keys are not reliable after 'project'

Penetration Resistant

- Control access to data
  - ensure users have restricted views of data
  - prohibit arbitrary free-data entry where possible
  - provide restricted-choice selections instead
- Don't forget: validate all inputs
  - #1 security error is failure to validate all inputs
  - SQL injection faults could kill the DB

injecting an extra SQL command in a simple text entry field

Enter student ID:

0011234567; drop table student

Lab 2: Read-only View

- Write the SQL for a view of the marks obtained by a student for studied modules
  - assume table Student (regNo, forename, surname)
  - assume table Module (modID, modName, lecturer)
  - assume table Study (regNo, modID, mark1, mark2)
- Outline syntax for a view
  - CREATE VIEW … AS SELECT … FROM … WHERE …;
  - no GROUP BY as we are not using any aggregate functions like COUNT(*) here
- We want to see in the resulting view
  - regNo, forename, surname, modID, modName, mark1, mark2

Updatable View?

Run a Poll

- Is this view updatable?

```
CREATE VIEW Transcript (regNo, surname, forename,
modID, modName, mark1, mark2) AS
  SELECT regNo, surname, forename, modID, modName,
    mark1, mark2 FROM (Student, Module, Study)
  WHERE  Student.regNo = Study.regNo
     AND  Module.modID = Study.modID;
```

Failsafe Transactions

- Transaction
  - a database transaction is a single, complete unit of work, which must either execute completely, or not at all
  - must be ACID: Atomic (all or nothing), Consistent (data integrity), Isolated (serializable), Durable (permanent)
- Protect against brief loss of service
  - wrap a set of updates in a transaction, if all updates must happen together (e.g. credit/debit in a money transfer)
  - the transaction must succeed, or fail, as a whole
  - upon failure, the database must rollback (forget temporary changes) or revert (recover the "before image" of the data)

Transaction Recovery

- Commit and rollback, or revert?
  - commit: make temporary changes permanent
  - rollback: discard temporary changes, before a commit
  - revert: restore DB to earlier state from saved old data
- Three possible strategies
  - deferred update: write changes to a temporary log file, commit all changes together, or rollback (discard) if incomplete (DB may have uncommitted changes)
  - immediate update: log the old value, change the main data, recover by reverting (extra update, but DB is consistent)
  - shadow paging: keep dynamic pointers to blocks of current, old data and merge or revert (needs memory management)

Concurrent Availability

- Concurrent access
  - multiple users access the same database simultaneously
  - transactions overlap, complete in nondeterminstic order
- Concurrency conflicts
  - dirty read – user1 reads data, as user2 is updating it, but has not committed; the data read is out of date
  - non-repeatable read – user1 runs a query twice, as user2 is modifying data; the query returns inconsistent data
  - phantom read – user1 runs a query twice, as user2 is adding to the data; the query returns extra phantom data
  - lost update – user1 and user2 make simultaneous updates to the same data; the last to commit is saved, the first update is lost

Concurrency Control - I

- Serialize all transactions
  - pro: all transactions forced to execute in sequential order
  - con: serial bottleneck reduces performance
- Row/page locking
  - pro: other transactions locked out while row is being updated
  - con: possible deadlock if transactions wait for each other
- Deadlock avoidance
  - pro: aborts transactions if all locks cannot be obtained
  - con: aborts and restarts too many valid transactions
- Deadlock detection
  - uses a wait-for graph to detect and break circular deadlocks, aborts an arbitrary transaction; results in fewer aborts

Concurrency Control - II

- Shared and exclusive locks
  - shared read-lock allows multi-user access for reading
  - exclusive write-lock prevents all further access while updating
- Phantom and intent locks
  - phantom lock reserves an empty row for later insertion
  - intent lock declares intention to commit changes, may co-exist with shared locks; becomes exclusive only during the commit phase
- Timestamping
  - each transaction is given a unique start-time and all rows are marked with their last-read and last-written timestamps
  - transactions can freely write any rows with earlier read/write stamps, and read rows with later read stamps; otherwise abort

Summary

- Authentication determines that a user is who they claim to be
- Authorisation determines what functions or data a user, or other software, has the right to use, access or update
- Integrity determines whether data has been tampered with during transit and whether it comes from a trusted source
- Confidentiality ensures that only the agreed parties can see data secured by encryption (shared key, or public-private key)
- Physical attacks and equipment failure are mitigated by distribution and data-replication
- Penetration attacks are mitigated by offering restricted views of data and strong input validation
- Operational failures are mitigated by transaction recovery
- Availability is assured partly through concurrent access