COMS3000/7003

Week 7

Access Control, Security Models, Information Theory

Assignment

- The journal links are down at present – UQ proxy is over quota – you currently have to **actually be on campus** (and go direct, don't use ezproxy) to look at these journals.
  - *IEEE Security & Privacy* and
  - *IEEE Computer*

- This affects everyone and everyone is equally disadvantaged, so I will not be considering individual extensions for this.

Certifications Reprised

i.e. where we have got to so far…

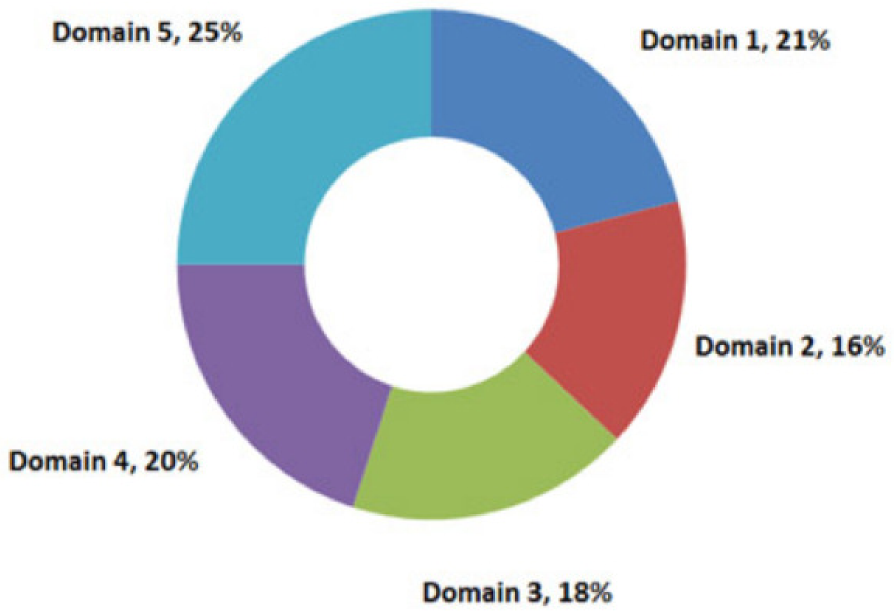
CISSP® Domains

- The CISSP CBK consists of 8 domains (used to be 10):
- **Security and Risk Management**
- **Asset Security**
- **Security Engineering (Physical Security)**
- **Communication and Network Security**
- **Identity and Access Management**
- **Security Assessment and Testing**
- **Security Operations**
- **Software Development Security**

ISACA

CISA Certified Information Systems Auditor® An ISACA® Certification

CISA Certification Job Practice Areas by Domain



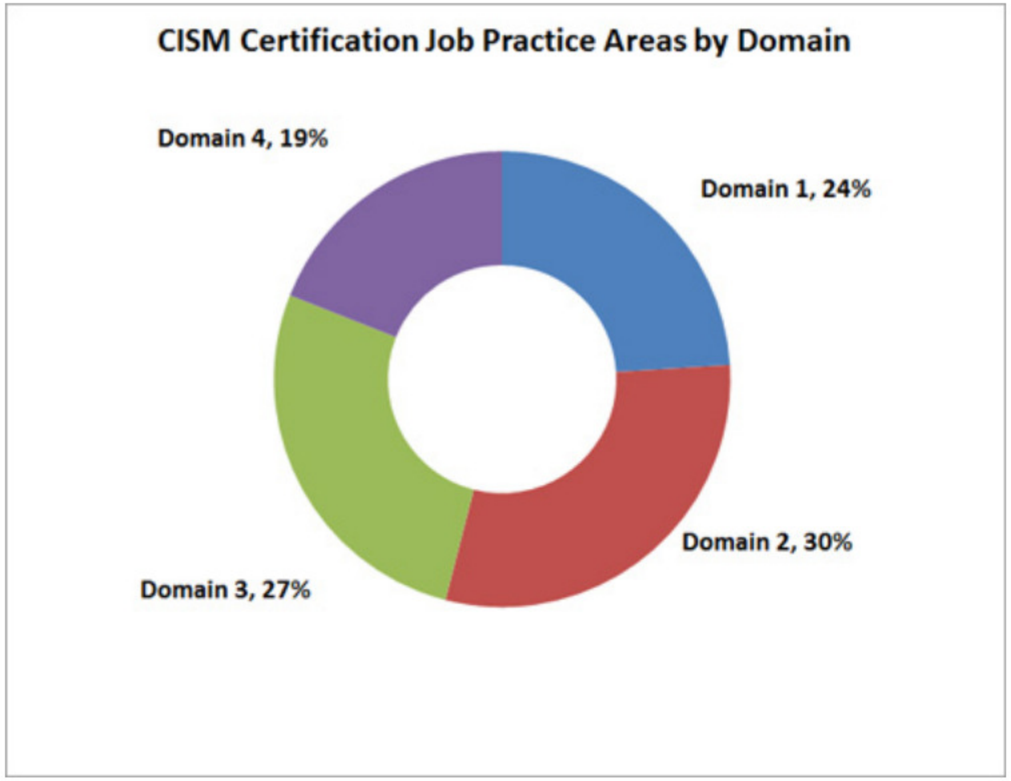
The job practice domains and task and knowledge statements are as follows:

Domain 1—The Process of Auditing Information Systems (21%)
Domain 2—Governance and Management of IT (16%)
Domain 3—Information Systems Acquisition, Development and Implementation (18%)
Domain 4—Information Systems Operations, Maintenance and Service Management (20%)
Domain 5—Protection of Information Assets (25%)

ISACA

Certified Information Security Manager

An ISACA Certification



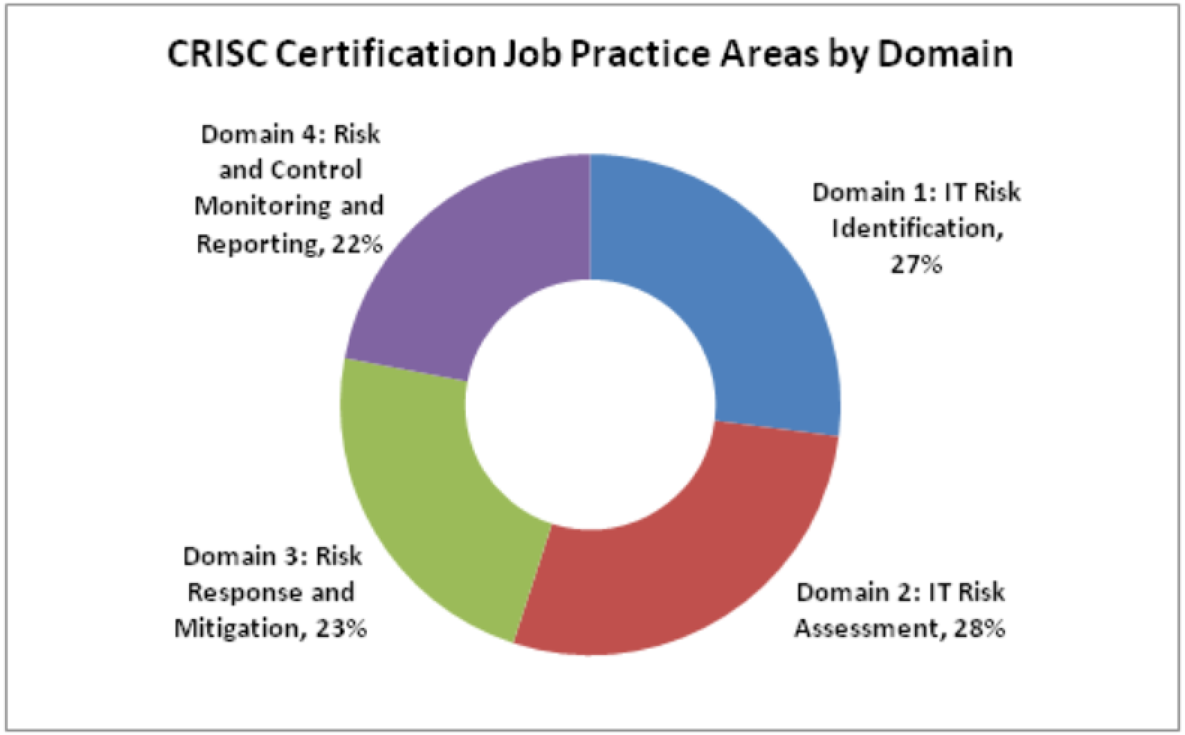
The job practice domains and task and knowledge statements are as follows:

Domain 1—Information Security Governance (24%)
Domain 2—Information Risk Management (30%)
Domain 3—Information Security Program Development and Management (27%)
Domain 4—Information Security Incident Management (19%)

ISACA

Certified in Risk and Information Systems Control

An ISACA® Certification



The job practice domains and task and knowledge statements are as follows:

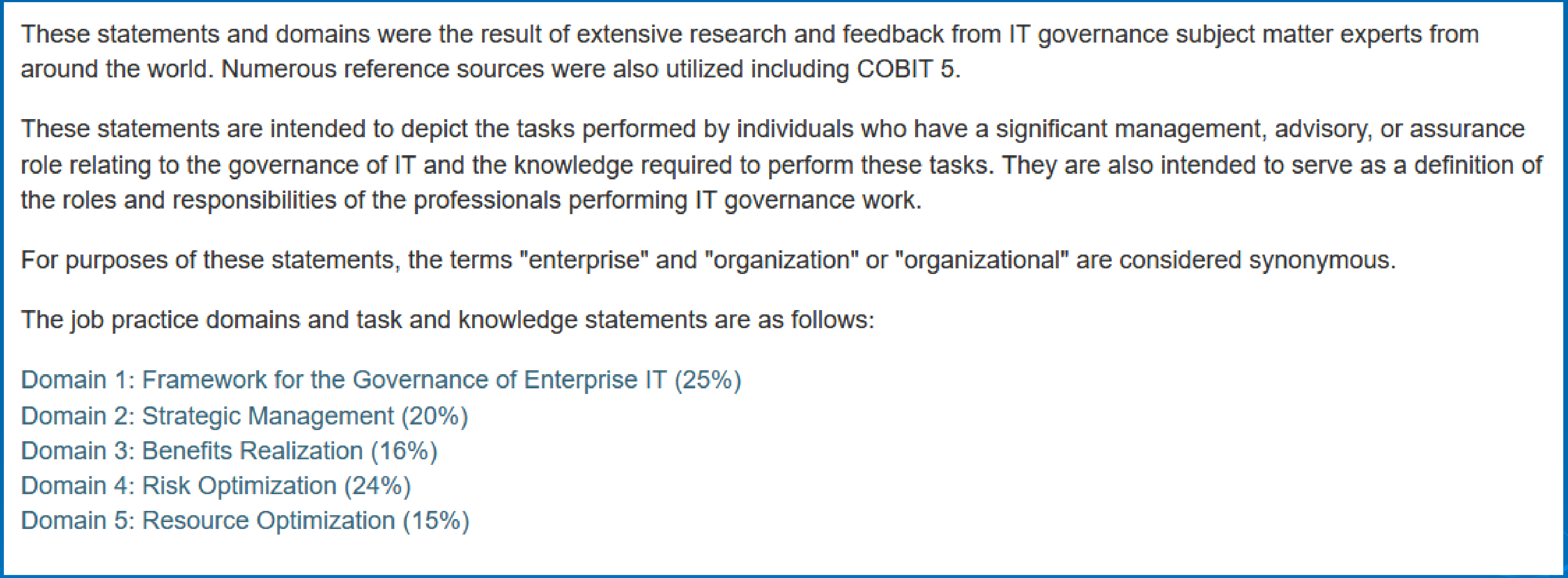
Domain 1—IT Risk Identification (27%)
Domain 2—IT Risk Assessment (28%)
Domain 3—Risk Response and Mitigation (23%)
Domain 4—Risk and Control Monitoring and Reporting (22%)

ISACA

Certified in the Governance of Enterprise IT®

CGEIT®

An ISACA® Certification



These statements and domains were the result of extensive research and feedback from IT governance subject matter experts from around the world. Numerous reference sources were also utilized including COBIT 5.

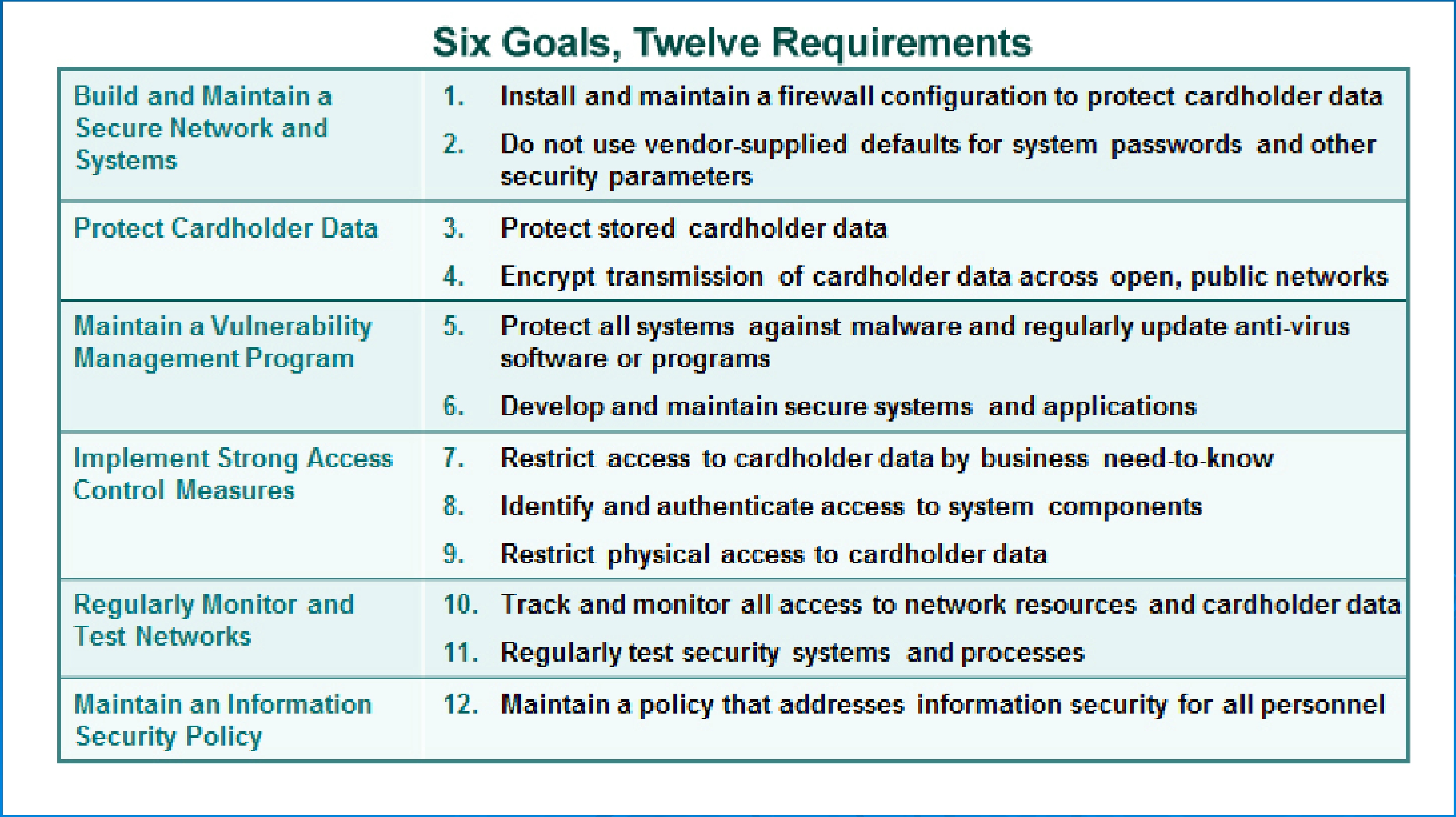
These statements are intended to depict the tasks performed by individuals who have a significant management, advisory, or assurance role relating to the governance of IT and the knowledge required to perform these tasks. They are also intended to serve as a definition of the roles and responsibilities of the professionals performing IT governance work.

For purposes of these statements, the terms "enterprise" and "organization" or "organizational" are considered synonymous.

The job practice domains and task and knowledge statements are as follows:

Domain 1: Framework for the Governance of Enterprise IT (25%)
Domain 2: Strategic Management (20%)
Domain 3: Benefits Realization (16%)
Domain 4: Risk Optimization (24%)
Domain 5: Resource Optimization (15%)

PCI Data Security Standard



Six Goals, Twelve Requirements

Build and Maintain a Secure Network and Systems	1. Install and maintain a firewall configuration to protect cardholder data
	2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	3. Protect stored cardholder data
	4. Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	5. Protect all systems against malware and regularly update anti-virus software or programs
	6. Develop and maintain secure systems and applications
Implement Strong Access Control Measures	7. Restrict access to cardholder data by business need-to-know
	8. Identify and authenticate access to system components
	9. Restrict physical access to cardholder data
Regularly Monitor and Test Networks	10. Track and monitor all access to network resources and cardholder data
	11. Regularly test security systems and processes
Maintain an Information Security Policy	12. Maintain a policy that addresses information security for all personnel

PCI Data Security Standard

- Reading:
- Review **Requirement 9** *Restrict physical access to cardholder data* of the PCI DSS:
- https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf
- And compare these requirements with Microsoft's controls in last week's lecture.

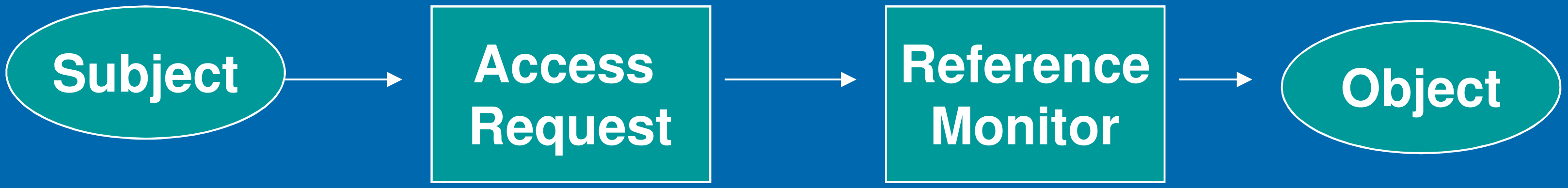
Any questions so far?

BACK TO…

COMPUTER BASED ACCESS CONTROL

Computer Based Access Control (Authorisation)

- We assume user has been authenticated
- How does an Operating System control access to resources (files, printer, memory, database record, network etc.)?
- Fundamental access control model:
  - Subject: active party
    - e.g. user, process
  - Object: passive party
    - e.g. file
    - Roles of object and subject depend on situation, can be reversed
      - Subject can become object and vice versa
  - Reference Monitor: grants or denies access



Access Control Matrix

- How can we specify an access control policy, i.e. *who can do what, with which objects*?
- We define the following:
  - set *S* of *subjects*
  - set *O* of objects
  - set *A* of *access operations*
- Access Rights can be defined as a **Access Control Matrix**
  - Entry M_{SO} specifies the set of access operations subject **s** can perform on object **o**

$$M = (M_{SO})_{s \in S, o \in O} \text{ with } M_{SO} \subset A$$

- Example:

	Bill.doc	Edit.exe	Fun.exe
Alice	-	{execute}	{execute, read}
Bob	{read, write}	{execute}	{execute, read, write}

Access Control Lists (ACL)

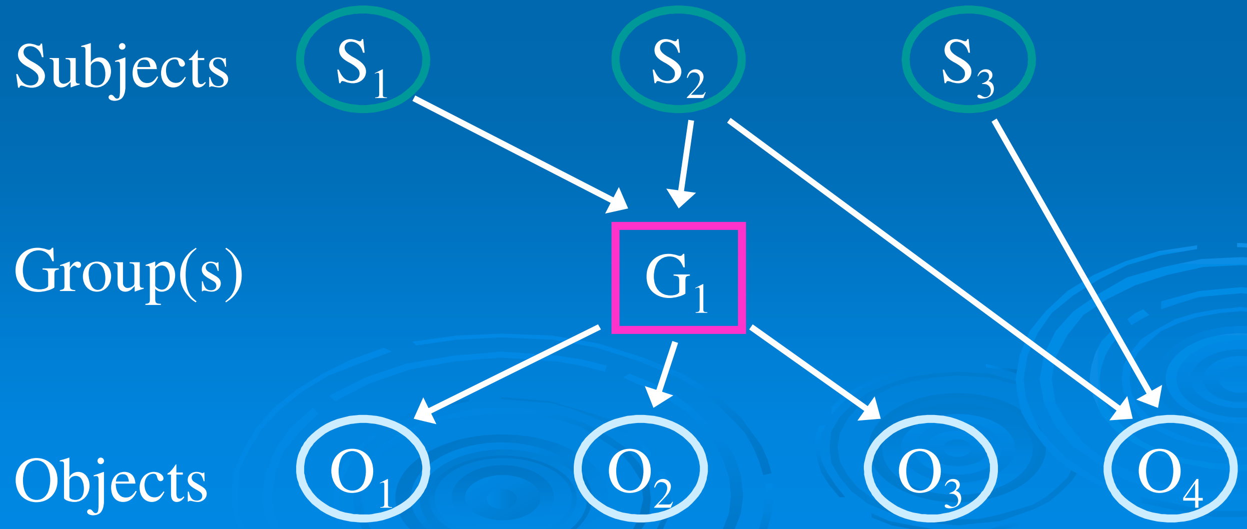
- Access rights are stored with objects
- Columns of Access Control Matrix (including user names in first column)

- Advantage:
  - Easy to see who has access to specific objects
- How about getting an overview of access rights of an individual users?
- How can access rights of an individual users be revoked?
  - Need to search through all ACLs → expensive operation

	Bill.doc	Edit.exe	Fun.com
Alice	-	{execute}	{execute, read}
Bob	{read, write}	{execute}	{execute, read, write}

Access Control Lists

- Managing access rights of a large number of individual users (subjects) via ACLs can be tedious
- Solution?
- Groups (or Roles)
  - Users (subjects) with similar access rights are aggregated in groups and access permissions are given to groups
  - Some policies allow membership to multiple groups, others not
  - Groups can be defined according to roles
    - e.g. in a Bank: teller, branch manager, branch accountant, …
  - Access rights can be revoked by removing a user from a group



$$\begin{tikzcd}
S_1 \arrow[dr] & & S_2 \arrow[dl] \arrow[ddrr] & S_3 \arrow[ddr] & \\
& G_1 \arrow[dl] \arrow[d] \arrow[dr] & & & \\
O_1 & O_2 & O_3 & & O_4
\end{tikzcd}$$

Unix Access Control

- Traditional Unix systems provide a limited form of ACLs
  - (“minimal ACLs”)
- The owner of each file (and root) can specify
  - Access rights (“permissions”) for the owner
  - Access rights for people in the same group
  - Access rights for everyone else (“other”, “world”)

- Each access right is a combination of
  - R – allowed to read the file
  - W – allowed to write to the file
  - X – allowed to execute the file

- Sample permission (9 bits for access rights):

```
-rwxr-x---   bill   students   thefile
```

  - First letter indicates if directory (d) or normal file (-)
  - **Letters 2-4: Owner** (Bill) can read, write, execute;
  - **Letters 5-7: Group** (student) can read & execute
  - **Letters 8-10: World** (everyone else) has no access
- Windows uses a similar system, with slightly more fine grained control

man chmod

- **"… A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. The first digit selects the set user *ID (4)* and set group *ID (2)* and restricted deletion or *sticky (1)* attributes. The second digit selects permissions for the user who owns the file: *read (4), write (2), and execute (1);* the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values."**

Read = 4, Write = 2, execute = 1

- Example:
- **chmod 640 file**
  - Read and write access for owner (6 = 4 + 2)
  - Read access for group
  - No access for other users

- Example:
  - **chmod 777 file**
  - Complete access for everybody

Unix Access Control

- In Unix, everything is treated as a file. (directories, interprocess communication channels, etc.)
- More recent versions of Unix (e.g. Linux) implement full ACL functionality
- For more information on ACLs in Linux
  - `man acl`

`DESCRIPTION`

- **`This manual page describes POSIX Access Control Lists, which are used to define more fine-grained discretionary access rights for files and directories…`**

More on Unix Access Control

- Files are not accessed directly by users, but by processes started by users
- Normally, a process has the ID and access rights of the user who starts it
- Example: Password Change
  - `/etc/shadow` contains hashes of passwords
    - Only writeable by *root*

```
uqport@moss:/etc$ ls -l shadow
-rw------- 1 root root 2585 Aug 27 21:37 /etc/shadow
```

- How can users change their own passwords, which requires updating the password hash?
  - Use the *passwd* program
  - But, if run by the user, *passwd* has typically no write access to the password file
- Solution?

SUID (SGID) Feature

- "Set User Id" (SUID) feature
  - (SGID → set group ID)
  - Executable files can be set to run, not with the user's ID (privileges), but with its owner's ID (privileges).
    - → SUID bit is set, indicated with letter 's' in place of 'x'
  - `-rwsr-xr-x    root    root     /usr/bin/passwd`
  - *passwd* executes with permissions of owner, i.e. root in this case
  - Has the rights to update `/etc/shadow` file
- Can you see any potential security problem with the SUID feature?

SUID Vulnerability

- Often, SUID programs are owned by *root*
  - → have unrestricted access
- Possible attack exploiting this?
  - Attacker can try to gain control of such a SUID program and make it execute arbitrary code, with unrestricted access
- How can this be done?
  - Complex programs are rarely bug free
  - Bugs can allow execution of arbitrary code
- Pre-eminent attack → "Buffer overflow Attack"
  - http://en.wikipedia.org/wiki/Buffer_overflow
- SUID can be dangerous if used carelessly

Principle of Least Privilege

- Very important general principle in information security
- Jerome Saltzer:
  - *"Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job."*
- Examples:
  - A process should give up super-user privileges, if no longer required, e.g. by resetting SUID bit as soon as possible.
  - It's a bad idea to be logged in as *root*, if not necessary to do the required job.

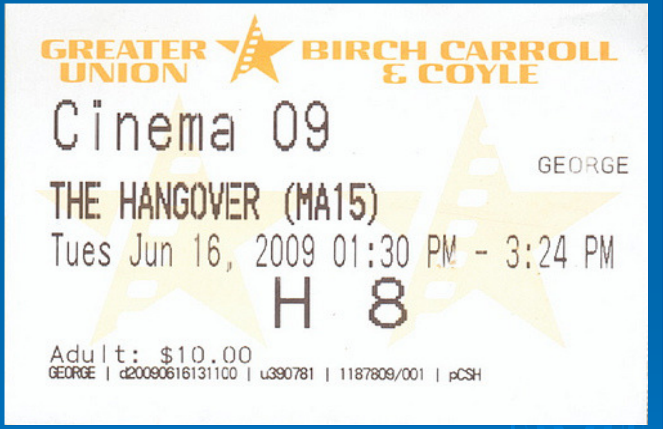
Some Homework

- Look up the history of the "sticky bit" on Unix executable files
- Does Linux do this for executable files?
- What does it do if set on a directory?

Any questions so far?

Access Control at the Movies

- How could you use ACLs to implement access control at the Movies?
  - Have a list of names of people who paid for a ticket for each movie session
  - Not very practical



GREATER UNION BIRCH CARROLL & COYLE

Cinema 09 GEORGE

THE HANGOVER (MA15)

Tues Jun 16, 2009 01:30 PM - 3:24 PM

H 8

Adult: $10.00

GEORGE | d20090616131100 | u390781 | 1187809/001 | pCSH

- Alternative: Tickets
  - Each person has ticket(s), which gives access to movie session(s)
- → Concept of 'Capabilities'

Capabilities

- Capabilities are an alternative to ACLs for managing access rights
- A **capability** is an (object – rights) pair
  - Used like a movie ticket
- Access rights are stored with subjects → Rows of Access Control Matrix
- Advantages:
  - Easy to see permissions of an individual users
  - Easy to revoke access for a particular user
- Disadvantages
  - Hard to see who has access to a particular object
  - Harder to revoke access to a particular object → must find all tickets
- Rarely used in practical systems
  - Mostly relevant in OS research
  - Example systems: Hydra, StarOS, Amoeba, L4, …

	Bill.doc	**Edit.exe**	**Fun.com**
Alice	-	{execute}	{execute, read}
Bob	{read, write}	{execute}	{execute, read, write}

Types of Access Control - Ownership

- Who has control of defining access rights?
- Who defines in UNIX/Windows the access rights to files?
  - Typically the owner of the files
  - → **“Discretionary Access Control” (DAC)**
    - i.e. “ at the discretion of the owner”
  - Exception: Which access control rule cannot be defined by the owner of a file in Unix?
    - Superuser (“root”) has generally unrestricted access

- Alternative
  - System-wide policy defines access to objects
  - → **“Mandatory Access Control”**
  - Often used in military and other high security contexts

Problems of DAC

- Suppose Bob has access to a *top secret* file *F*. Trudy does not have access to *F* but would like to.
- What can Trudy do to gain access to the file?
  - She can ask Bob to send it to her or give her access permission
    - ("Assuming Bob collaborates")
  - She can trick Bob and the system into given her access. How?
  - She can write a program that does two things
    - It executes the following sequence of commands
      - 1) Create a new file F2
        2) Grant Bob write access to F2
        3) Grant Trudy read access to F2
        4) Copy content of F to F2
    - Trudy sends Bob the program and tells him he should try it, e.g. pretending it is a new game.
    - When Bob runs the program, it has all of Bob's access rights and executes the above 4 commands .
      - → Trudy gets access to the content of *F*, now stored in *F2*

Problems of DAC

- The problem is that programs (and users) cannot be trusted
  - Especially not those programs downloaded from untrusted sources
  - The program may perform additional functions unknown to the user
- This type of program (malware) is referred to as a …?
  - *Trojan (Trojan horse)*
- Discretionary Access Control is insufficient to protect against these kind of attacks
  - programs cannot be analysed for all possible unexpected behaviours
- DAC cannot prevent users with access to highly sensitive information from sharing it with others, voluntarily or involuntarily.
- → In high a security context, e.g. Military, Government, **Mandatory Access Control** is used

Mandatory Access Control (MAC) – Multi-level Security

- Each file (object) has a **classification**, i.e. a label indicating its sensitivity level
  - e.g. Confidential → Secret → Top-secret
- Each user (subject) has a **clearance**, indicating the sensitivity level of objects he/she has access to
  - e.g. Confidential → Secret → Top-secret

- A user can only access an object if his/her clearance is equal or higher than the classification of the object
  - We say, the clearance "dominates" the classification

- Classifications and Clearances are assigned by the system/management. Users have no discretion about who can access which information.
  - → Mandatory Access Control (MAC)
  - Also known as "Multilevel Security"

Any questions so far?

Classification/Clearance Example

- Classifications:
  - *unclassified* $\rightarrow$ *classified* $\rightarrow$ *secret* $\rightarrow$ *top-secret*

- Alice has clearance "*secret*"
- Which documents can she read?
  - *Unclassified, classified, secret*

- Bob has clearance "*classified*"
- He has access to:
  - *unclassified* and *classified* documents

Categories - Compartments

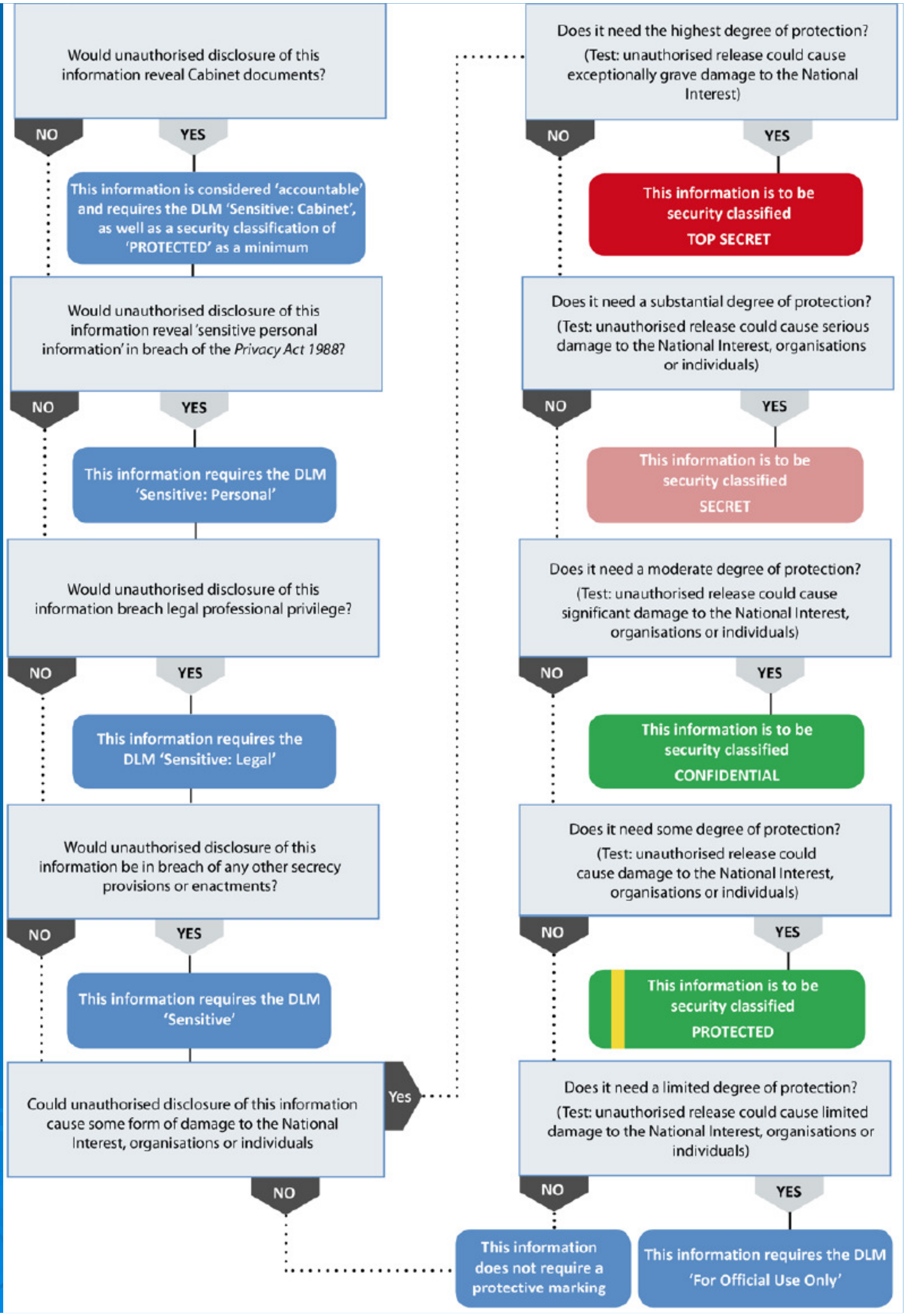
- Somebody with clearance "secret" has access to all documents of classification "secret"
  - This is often not necessary
- Principle of Least Privilege, or "Need to know"
  - Users should only have access to information necessary to do their job
- Example:
  - Documents on nuclear weapons and latest cryptography are both classified as top-secret
  - A user with a clearance of "top-secret" working on cryptography does not need and should not have access to sensitive information about nuclear weapons
- Solution:
- → Categories within classification levels
  - Known as *compartments*

Compartments

- Example:
  - Classifications:
    - secret, top-secret
  - Compartments:
    - Nuclear, Crypto
- Users only have access to their level of classification within their assigned compartments
- Allows more fine grained access control
- → implementation of "Need to know"

The Australian Government Security Classification System (AGSCS)

https://www.protectivesecurity.gov.au/informationsecurity/Documents/INFOSECGuidelinesAustralianGovernmentSecurityClassificationSystem.pdf



Would unauthorised disclosure of this information reveal Cabinet documents?

This information is considered 'accountable' and requires the DLM 'Sensitive: Cabinet', as well as a security classification of 'PROTECTED' as a minimum

Would unauthorised disclosure of this information reveal 'sensitive personal information' in breach of the *Privacy Act 1988*?

This information requires the DLM 'Sensitive: Personal'

Would unauthorised disclosure of this information breach legal professional privilege?

This information requires the DLM 'Sensitive: Legal'

Would unauthorised disclosure of this information be in breach of any other secrecy provisions or enactments?

This information requires the DLM 'Sensitive'

Could unauthorised disclosure of this information cause some form of damage to the National Interest, organisations or individuals

Does it need the highest degree of protection? (Test: unauthorised release could cause exceptionally grave damage to the National Interest)

This information is to be security classified TOP SECRET

Does it need a substantial degree of protection? (Test: unauthorised release could cause serious damage to the National Interest, organisations or individuals)

This information is to be security classified SECRET

Does it need a moderate degree of protection? (Test: unauthorised release could cause significant damage to the National Interest, organisations or individuals)

This information is to be security classified CONFIDENTIAL

Does it need some degree of protection? (Test: unauthorised release could cause damage to the National Interest, organisations or individuals)

This information is to be security classified PROTECTED

Does it need a limited degree of protection? (Test: unauthorised release could cause limited damage to the National Interest, organisations or individuals)

This information does not require a protective marking

This information requires the DLM 'For Official Use Only'

SELinux

- *"NSA **Security-enhanced Linux** is a set of patches to the Linux kernel and some utilities to incorporate a strong, flexible mandatory access control (MAC) architecture into the major subsystems of the kernel. …"*
  - Integrated into the Linux kernel since version 2.6, 2003.

- https://en.wikipedia.org/wiki/National_Security_Agency
- https://en.wikipedia.org/wiki/Security-Enhanced_Linux

Any questions so far?

Security Models

Lattice-based Access Control Model

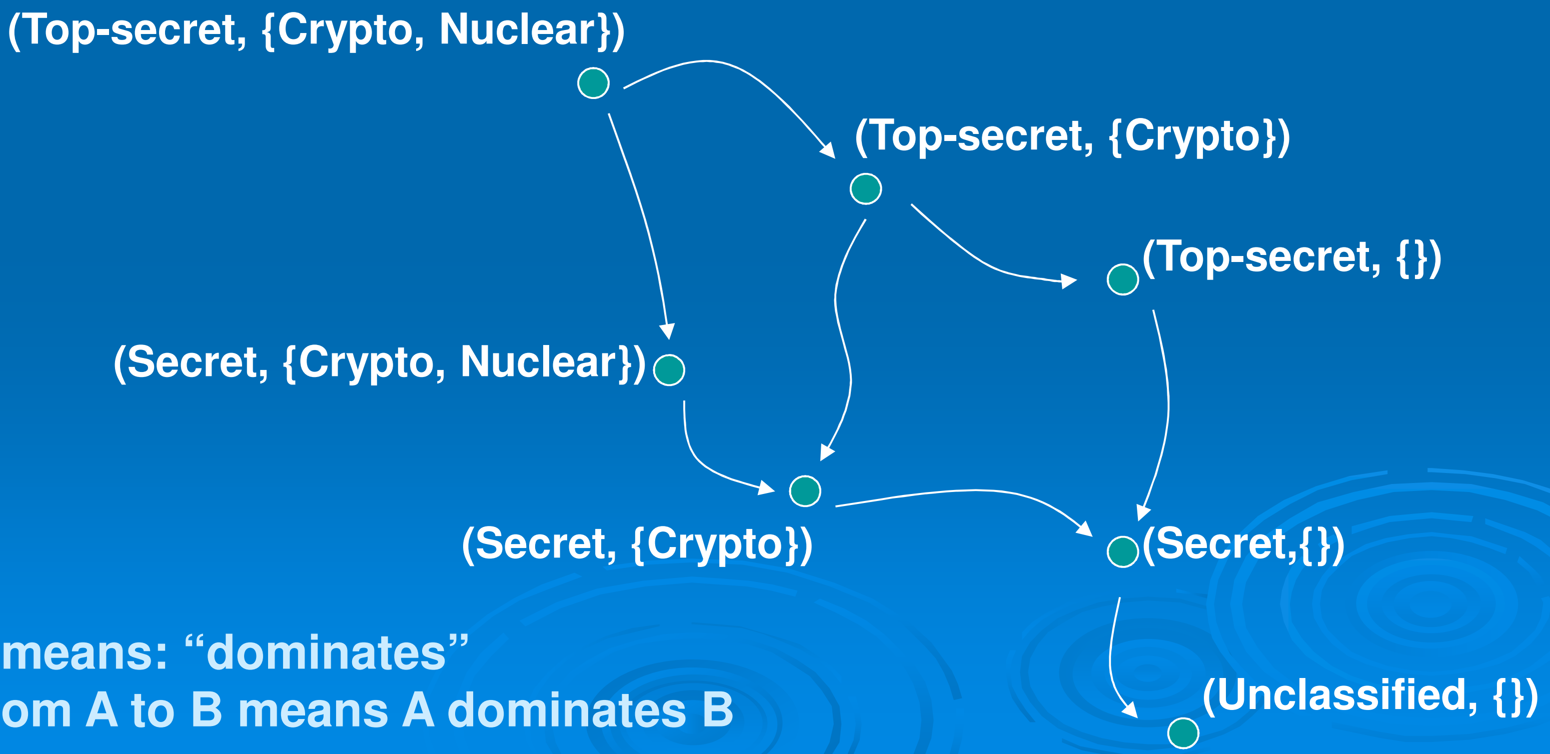
- Denning, Dorothy E. "A lattice model of secure information flow."*Communications of the ACM* 19.5 (1976): 236-243.
- Elements of a Lattice Model
  - An ordered sequence of sensitivity **levels**
  - A set of **compartments** (sometimes called *codewords*)
  - A “dominates” relation, written ≥
- Nodes in a lattice consist of a sensitivity label and a set of compartments
- Example:
  - {secret, Nuclear}, {secret, Crypto}, {secret}, {top-secret, Nuclear}
- Some of these are comparable, others are not
- Example:
  - {top-secret} ≥ {secret}
  - {top-secret, Nuclear} ≥ {secret, Nuclear}
  - {top-secret, Crypto} ?? {secret, Nuclear}
    - No dominance relationship
- To have access to an object, a subject needs to have the necessary **clearance** AND **compartment**.

Properties of a Lattice

- A, B are nodes in a lattice, i.e. classification, compartment-set pair)
  - $A \geq B$ & $B \geq C \Rightarrow A \geq C$ (transitivity)
  - $A \geq A$ (reflexivity)
  - $A \geq B$ & $B \geq A \Rightarrow A = B$ (antisymmetry)
- Partial ordering
  - Not every two elements need to be comparable
  - For every A and B, there exists a greatest lower bound
  - For every A and B, there exists a least upper bound

Lattice - Example

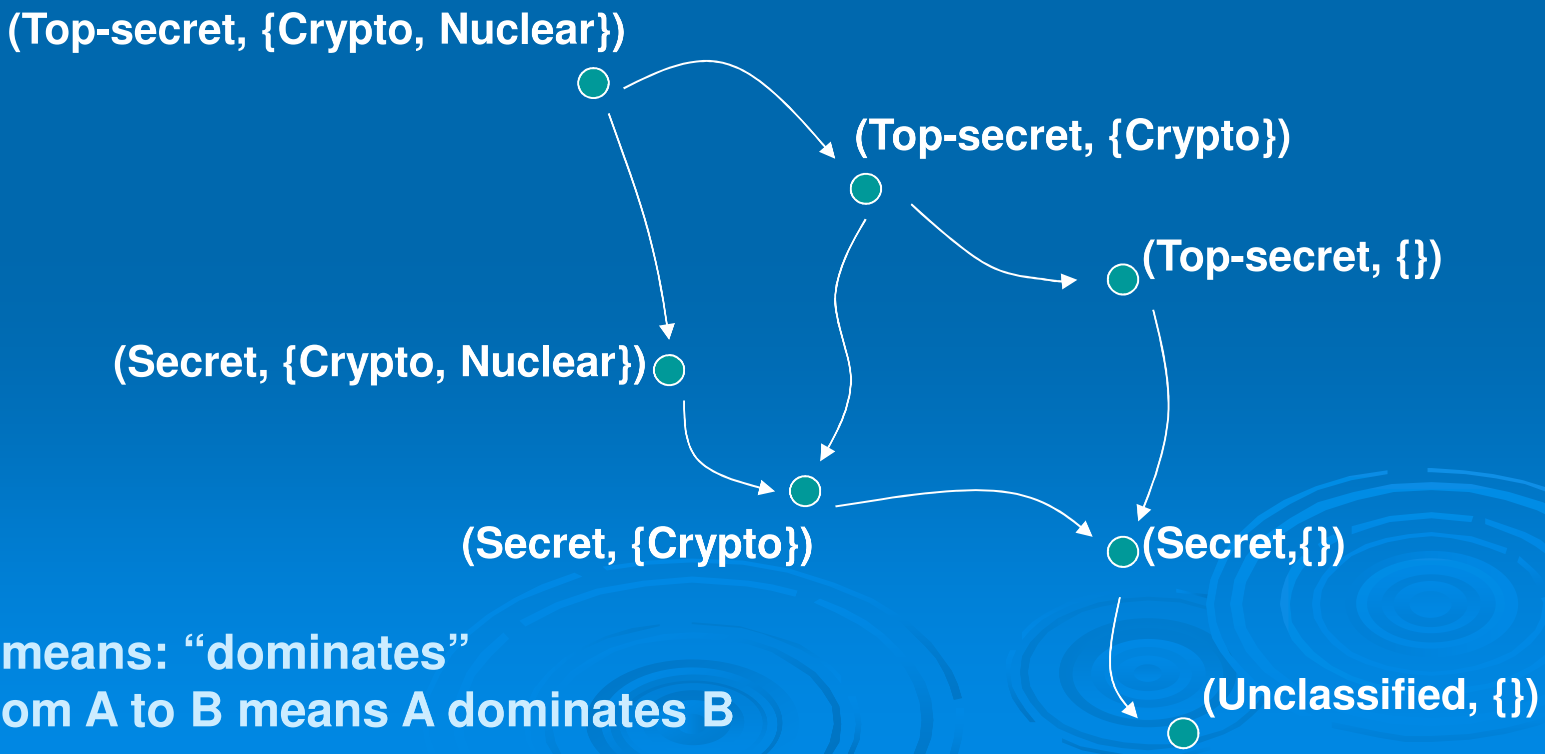
- Example:
  - Classifications: unclassified, secret, top-secret
  - Compartments: Nuclear, Crypto
- We have the following nodes:
  - (top-secret, {Crypto, Nuclear}), (top-secret, {Crypto}), (top-secret, {}), (secret, {Crypto, Nuclear}), (secret, {Crypto}), (secret, {}), (unclassified,{})
- Lattice:



Arrow means: "dominates"
Path from A to B means A dominates B

Lattice - Example

- Can Bob with classification/compartment (Top-Secret, {}) read a document with (Secret, {Crypto})?
  - No. There is no corresponding path in the lattice.



Arrow means: "dominates"
Path from A to B means A dominates B

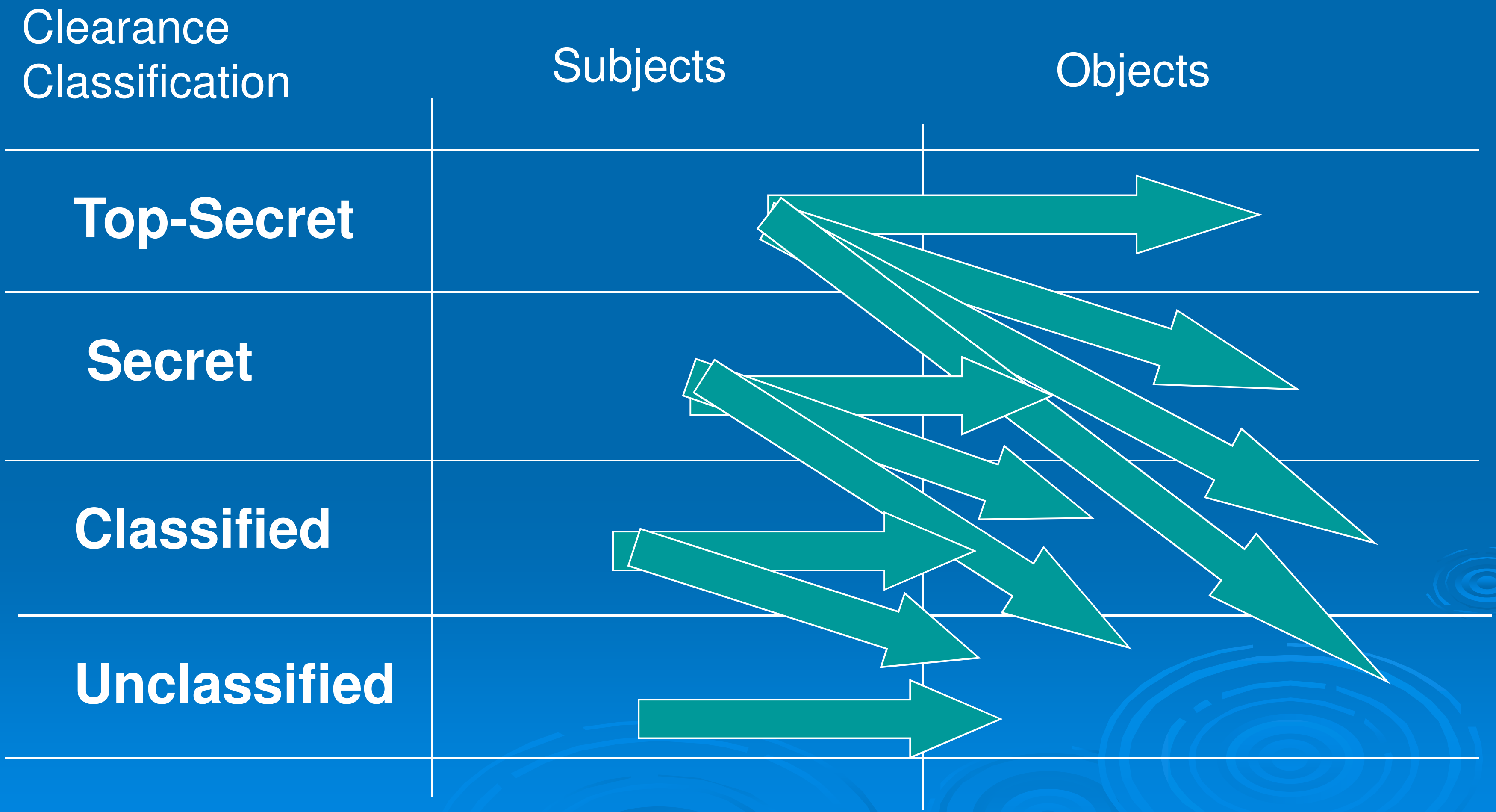
Bell-LaPadula (BLP) Model

- Goal:
  - A formal (mathematical) model of a security policy
  - For Government and Military applications
- Developed in early 1970s, funded by DoD in the US
- Motivation: multi-user systems
  - Alternative: Separate machines for each level of classification
- Each user has a clearance
- Each process operating on behalf of a user is a subject
  - A process's clearance may be lower than its owner's clearance if the user wants it to be
    - → "principle of least privilege"
- Each file is an object, and hence has a classification.
- BLP is concerned about confidentiality
- It tries to prevent information from leaking from a high level to a lower level
  - e.g. users with clearance 'secret' should not see any documents with label 'top-secret'

Bell-LaPadula

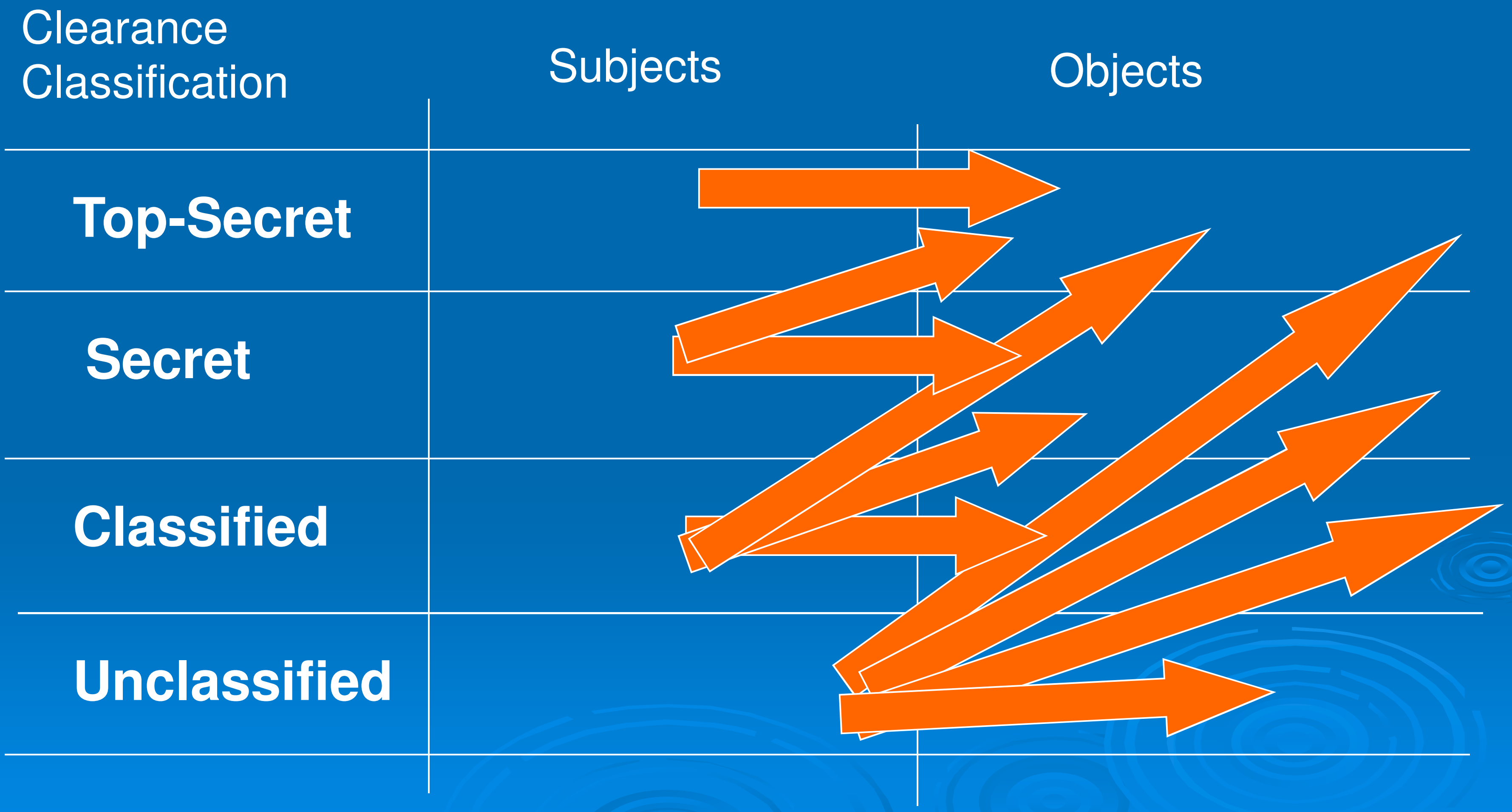
- **Two Rules to prevent information from flowing "downwards"** (with admittedly silly names):
  - **"Simple Security Property"**: A subject can only **read** an object if its clearance dominates the object's classification.
    - "No Read Up"
    - Is that enough? Does it guarantee that no information flows downwards?
    - No. User (or Trojan horse) with Top-security clearance could copy a top-security file to a new file with label "unclassified"
  - ***(star)-property**: a subject at a given security level must not write to any object at a lower security level
    - "No Write Down"

BLP Read Access
ss- Property ("No read up")



BLP Write Access

*-Property ("No write down")



High Water Mark

- Consider a subject with a secret clearance
- The subject has not accessed any files yet
  - No need to be at the highest level
  - Its *current* security level is unclassified
- This subject can write to an unclassified file
- The subject now reads a classified file
- Its current level increases to classified
  - → It can no longer write to an unclassified file

- Can the subject's security level be decreased to unclassified again?
  - No. This would allow him/her to read sensitive data and then write it to a lower level

- The security level of subjects can only be increased
- We say the current security level is a **High Water Mark**
  - Can only go up

BLP - Discussion

- Significant security model
  - Important for design of new secure operating systems
  - Implemented in Multics
    - OS developed by MIT in 1960s
    - “predecessor” on Unix
- Limitations:
  - Only addresses confidentiality (e.g. not integrity)
  - Does not address the problem of managing classifications
    - How are classifications assigned, changed?
    - How are clearances assigned, changed?
  - Not very practical for standard applications, complex
  - Does not address the problem of **covert channels**
    - ‘Unintended’ communication channel
    - e.g. users can communicate via ‘modulating’ CPU load
    - http://en.wikipedia.org/wiki/Covert_channel

Security Evaluation/Certification "Common Criteria"

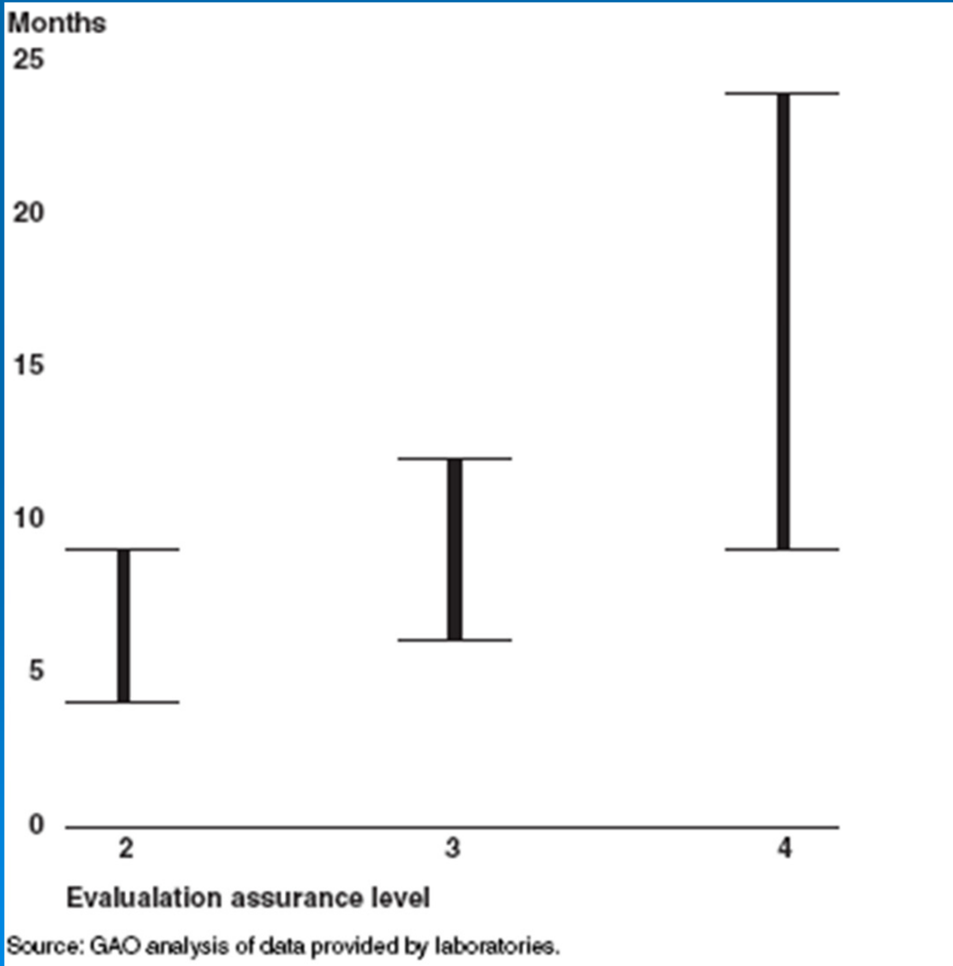
- Common Criteria for Information Technology Security Evaluation (Common Criteria or CC)
  - international standard (ISO/IEC 15408) for evaluation and certification of security of computer systems
    - Functional and in terms of Assurance Level
      - Evaluation Assurance Level (EAL), EAL1 (lowest) – EAL7 (highest)
  - Allows specification of security requirements of a system, and rigorous evaluation against them.
  - Largely used by Government Agencies

- Example
  - Windows 7 is certified at EAL4 (not Win 10 – just PP compliant)
  - EAL4: Methodically Designed, Tested, and Reviewed

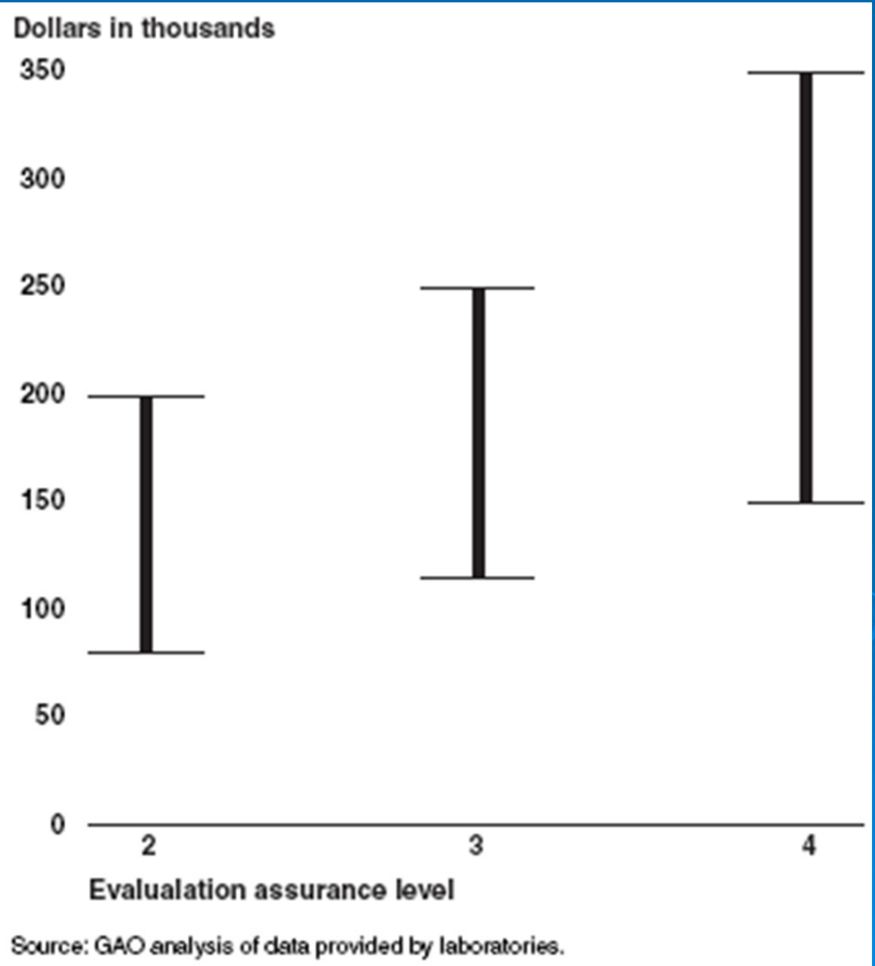
- https://www.commoncriteriaportal.org/
- https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf

Common Criteria Certification

- Certification is generally a lengthy and costly process



Source: GAO analysis of data provided by laboratories.



Source: GAO analysis of data provided by laboratories.

Any questions so far?

Basics of Information Theory

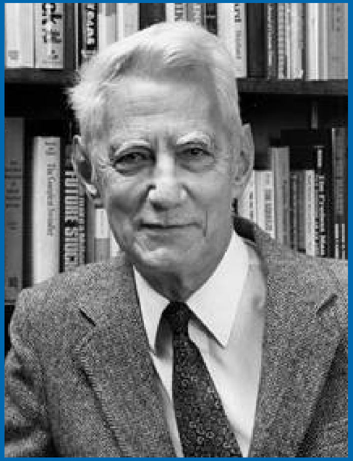
What is Information?

- The name of this course is “Information Security”
- What is information and how can we quantify it?
- Example:
  - A: I’m telling you that tomorrow, the weather will be fine.
  - B: I’m telling you that there will be a major hailstorm tomorrow.
- In which case do I give you more information?
  - Intuitively, in scenario B.
  - But, why, and by how much?
- We want a mathematical model/definition of information

Claude Shannon

- Claude Shannon (1916 – 2001)
  - “The Father of Information Theory” and “Father of the digital/information age”
  - I~~nventor~~ first publication of the term “bit”
  - Provided mathematical foundation of electronic communication and cryptography
    - E.g. ‘Shannon Limit ‘of communication channels
- Also famous for his ‘non-academic’ inventions
  - Motorized pogo stick
  - Juggling robot
  - The “Ultimate Machine”
    - https://www.youtube.com/watch?v=cZ34RDn34Ws

Shannon's Definition of Information



- Claude Shannon, 1916 – 2001
  - Two most significant publications (while at Bell Labs)
    - A Mathematical Theory of Communication (1948)
    - Communication Theory of Secrecy Systems (1949)
- Shannon's definition of Information:
  - "Information is reduction of uncertainty"
  - Model:
    - Information is gained by observing the outcome of a random experiment.
    - Random variable X with a finite set of values (outcomes) $x_1, x_2, \ldots x_n$
  - For example:
    - Before tossing a coin, you are uncertain about the outcome.
    - After the toss, there is no uncertainty left about the outcome.
    - The amount of information you gained by observing the outcome is equal to the amount of uncertainty that was reduced.
      - X: outcome of coin toss, x_1="Heads", x_2="Tails"

Measure of Information

- The unit of Information is *bits* (binary digits)
  - Term introduced by Shannon in his 1948 paper
- *I(X) is the average amount of Information gained from observing the outcome of the random variable X.*
  - *or*
- *I(X)* is also the minimum average number of bits needed to encode all possible outcomes of X
  - or the content of message with the outcome of X
- These two definitions are equivalent.

Information - Entropy

- Often, Shannon Information is also called ***"Entropy"***
- Story behind the name Entropy
  - Claude Shannon, as quoted in
    - M. Tribus, E.C. McIrvine, "Energy and information", *Scientific American*, 224, 1971
  - *"My greatest concern was what to call it. I thought of calling it 'information', but the word was overly used, so I decided to call it 'uncertainty'.*
  - *When I discussed it with John von Neumann, he had a better idea. Von Neumann told me, 'You should call it entropy, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it already has a name.*
  - *In the second place, and more important, nobody knows what entropy really is, so in a debate you will always have the advantage."*

Measure of Information

- How much information (in bits) do we get (on average) by observing the outcome of a random experiment with N possible outcomes?
  - For now, we assume that all outcomes are equally likely

- Tossing a coin (N=2)
  - I(X) = 1 bit
- Rolling a 4-sided dice (N=4)
  - I(X) = 2 bits
- Rolling an 8-sided dice (N=8)
  - I(X) = 3 bits

- Rolling an N-sided dice
  - I(X) = ?
  - $I(X) = \mathbf{\log_2 N}$ bits

- We could use a different base of the logarithm, which would give us a different unit of information (rarely used)
- $I(X) = \mathbf{\log_e N}$ nats (natural unit)
- $I(X) = \mathbf{\log_{10} N}$ dit (decimal digit) or Hart (Hartley)

Another Example

$I(X) = \mathbf{\log_2 N}$ bits

- How much information do we get by learning the numbers of Gold Lotto?
  - Pick 6 numbers, from 1 to 45 (let's ignore supplementary numbers)
  - N=?
    - N = 8,145,060
      - Number of Combinations
    - I(X)=?
  - $I(X) = \mathbf{\log_2}$ (8,145,060) = 22.96 bit

$$\frac{n!}{r!(n-r)!} = \binom{n}{r}$$

choose r our of n

- This information would obviously be worth much more before the draw. However, information theory is not concerned with the *value* of information, only its *quantity*.

Measure of Information

- In general, if we are observing a random experiment with N equally likely outcomes of the random variable X:
  - $I(X) = \log_2 N$
- What is the probability *p* for each of the N outcomes ?
  - $p = 1/N$
  - $N = 1/p$
  - Now we can rewrite the definition of information:

$$I(X)=\log_2 N = \log_2(1/p) = -\log_2(p)$$

We will use this version from now on …

Information vs. Encoding

- How can we encode the outcome of a coin toss?
  - “Heads”, “Tails”
  - “H”, “T”
  - “0”, “1”
  - “the result is Heads”, “the result is Tails”
  - “0000000000000000”, “1111111111111111”
  - ….
- The amount of information in this example is 1 bit, irrespective of the way we encode it.
- Information I(X) is the minimal number (lower bound) of bits we need to encode information.
  - In the above example, “0”, “1” is such an optimal encoding, using only a single bit

Coding

- Random variable X with N=8 equally likely outcomes
  - I(X)=?
  - I(X) = $\log_2 8$ = 3 bits
  - We gain 3 bits of information by observing the outcome
  - A minimum of 3 bits is used to encode all possible outcomes
- Random variable X with N=3 outcomes or ‘symbols’ {A,B,C}
  - I(X) = ?
  - I(X) = $\log_2 3$ = 1.58 bits
- How can we encode this in messages of ‘0’s and ‘1’s?
- First approach:
  - A → 00
  - B → 01
  - C → 10
  - 11 unused
- 2 bits per outcome or symbol
- According to Shannon, we should be able to do better than this. How?

Coding

- Second approach:
  - We use 'variable length' codes
    - We don't have any separator to delimit the different code words.
  - Let's try this:
  - A → 0
  - B → 1
  - C → 10
  - If we encode ABC we get:
    - 0110

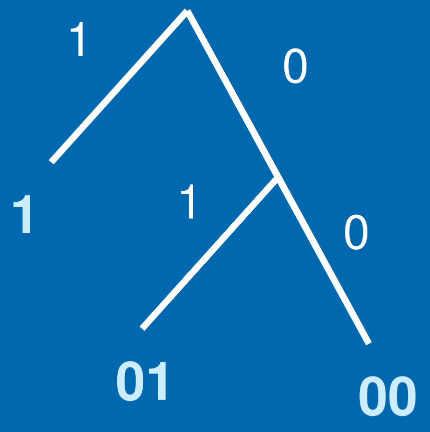
- Now let's go backwards and try to decode 0110:
  - ABC
  - But also ABBA
- The code does not allow unambiguous decoding
- What do we need to change?
  - We need a so-called prefix-free code
  - No code word can be the prefix of another code word
    - → No ambiguity
  - In the example above, 1 (B) is the prefix of 10 (C)

Prefix-free Codes

- Are telephone numbers (which are variable length) an example of a prefix-free code? (phone numbers range from 4 to 15 digits)
  - Yes
  - Imagine if that was not the case. For example, imagine the following two numbers were valid phone numbers:
    - 33445566
    - 3344556
  - → number 33445566 would never be reached

Prefix-free Binary Code

- We can create a prefix-free binary code by selecting the code words as the leaves of a binary tree
- Let's try again:
  - A → 1
  - B → 01
  - C → 00
- If we encode ABC we get:
  - 10100



- Now let's go backwards and decode 10100:
  - ABC
  - No ambiguity!
- Average code word length? (Assuming all symbols are equally likely)
- Weighted average of code words:
  - 1/3 * 1 + 1/3 * 2 + 1/3 * 2 = 1.666…
- 'Huffman coding' is a mechanism to find optimal prefix-free codes
  - Idea: Use short code words for frequent symbols, and longer ones for less frequent symbols
  - Used for lossless compression
- The theoretical minimal average code length (bits) is the Entropy I(X).
  - Lower bound

Measure of Information

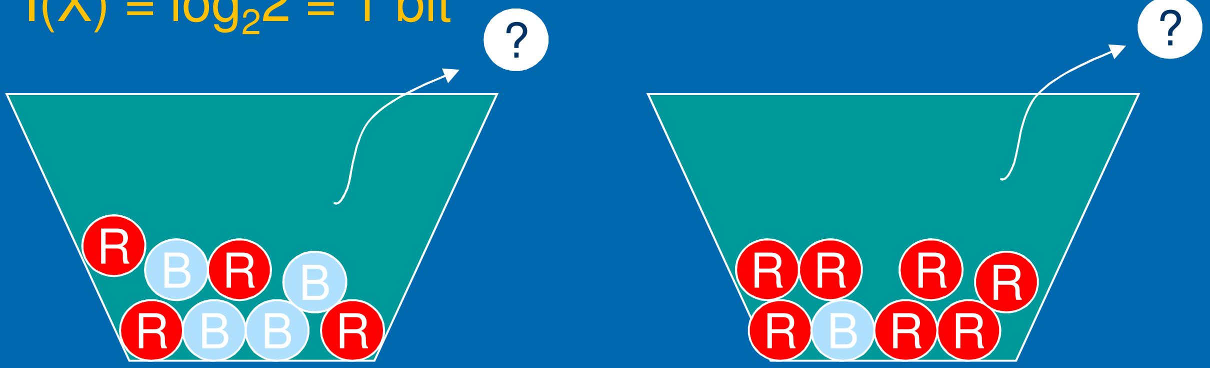
- So far, we have assumed that all outcomes or messages are equally likely, with p= const = 1/N
  - **$I(X) = \log_2(N) = \log_2(1/p) = -\log_2(p)$**

- This is a definition of Information given by Ralph Hartely, 20 years before Shannon's famous definition

- We are not quite there yet….

Measure of Information

- Consider the following random experiment:
  - Randomly draw a ball from the urn. Outcomes = {red, blue}
  - How much information does that give us according to our current (Hartley's) definition?
    - $I(X) = \log_2 2 = 1$ bit



- Consider the urn on the right:
  - Do we get the same amount of information if we draw red or blue?
  - We sort of expect the outcome to be 'red' and are not very surprised if this is the case. It is intuitively clear we get less information than when blue is drawn.
    - Intuitively, the amount of information corresponds to the level of surprise
  - Similar, but more extreme example:
    - Red ball: you did NOT win Division 1 price in Gold Lotto
    - Blue ball: you did win Division 1 price in Gold Lotto

Measure of Information

$I(X) = \log_2(N) = -\log_2(p)$

- The weakness of Hartley's definition of Information is that it ignores the different probabilities of the outcomes.
- How can we fix it?
- Choosing a **blue** ball from the second urn is in a sense the same as choosing 1 out of 8 possibilities ($p_1=1/8$). In this case we have:
  - $I(X) = ?$
  - $I(X) = \log_2 8 = -\log_2(1/8) = 3$ bits
- Choosing a **red** ball from the second urn is in a sense like choosing 7 out of 8 possibilities ($p_2=7/8$).
  - $I(X) = ?$
  - $I(X) = -\log_2(7/8) = 0.19$ bits

- What's the **average** information we gain by drawing a ball from this urn?
  - Weighted average of 'Hartley Information':
    - For each outcome, multiply the information gained with its probability
    - Add up for all outcomes
  - I(X) = Probability(blue) * I(blue) + Probability(red) * I(red)
  - $I(X) = p_1*(-\log_2(p_1)) + p_2*(-\log(p_2)) = 1/8 * 3 \text{ bits} + 7/8 * 0.19 \text{ bits} \approx 0.54$ bits

Shannon's Measure of Information

$I(X) = -\log_2(p)$

- We can write the result from the previous example for the general case.
- Here, our random experiment has N outcomes with probabilities $p_1, p_2, \ldots p_N$
- Shannon Information (Entropy) is the weighted average of Hartley Information:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

- When all outcomes are equally likely, Hartley information is the same as Shannon Information
- From now on, we will use **H(X)** for Shannon's definition of Information, or Entropy

Time for an Example Class Exercise

$\log_2(1/p) = -\log_2(p)$

- You can work in teams of 2-3
- Bob sends messages to Alice. The messages are either A, B, C, or D with the following probabilities (relative frequencies):
  - A: 50% (p_1=0.5)
  - B: 25% (p_2=0.25)
  - C: 12.5% (p_3=0.125)
  - D: 12.5% (p_4=0.125)
- What is the Entropy (Shannon Information) of Bob's messages?

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Class Exercise, Solution

- A: 50% = ½ = p_1
- B: 25% = ¼ = p_2
- C: 12.5% = 1/8 = p_3
- D: 12.5% = 1/8 = p_4

- Use: **log(1/x) = -log x**

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i =$$
$$-(p_1 \log_2 p_1 + p_2 \log_2 p_2 + p_3 \log_2 p_3 + p_4 \log_2 p_4) =$$
$$-(1/2 \log_2 (1/2) + 1/4 \log_2 (1/4) + 1/8 \log_2 (1/8) + 1/8 \log_2 (1/8)) =$$
$$1/2 \log_2 (2) + 1/4 \log_2 (4) + 1/8 \log_2 (8) + 1/8 \log_2 (8)) =$$
$$0.5 + 0.5 + 3/8 + 3/8 = 1.75 bits$$

Solution, Alternative Method

- A: 50% = ½ = p_1
- B: 25% = ¼ = p_2
- C: 12.5% = 1/8 = p_3
- D: 12.5% = 1/8 = p_4

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i$$

	p	$-\log_2 p$	$- p^* \log_2 p$
A	0.5	1	0.5
B	0.25	2	0.5
C	0.125	3	0.375
D	0.125	3	0.375
Total			1.75 bits

Binary Entropy Function

- We have a random variable X with 2 outcomes x1 and x2, with probabilities p and (1-p).
- For which value of p do we have maximum Entropy/Information?

- We have maximum Entropy/Uncertainly if both outcomes are equally likely
  - → 1 bit
  - See Tutorial

- In general, for a random variable with N outcomes, we have maximum Entropy if all outcomes are equally likely.

H(X) as a function of p1.

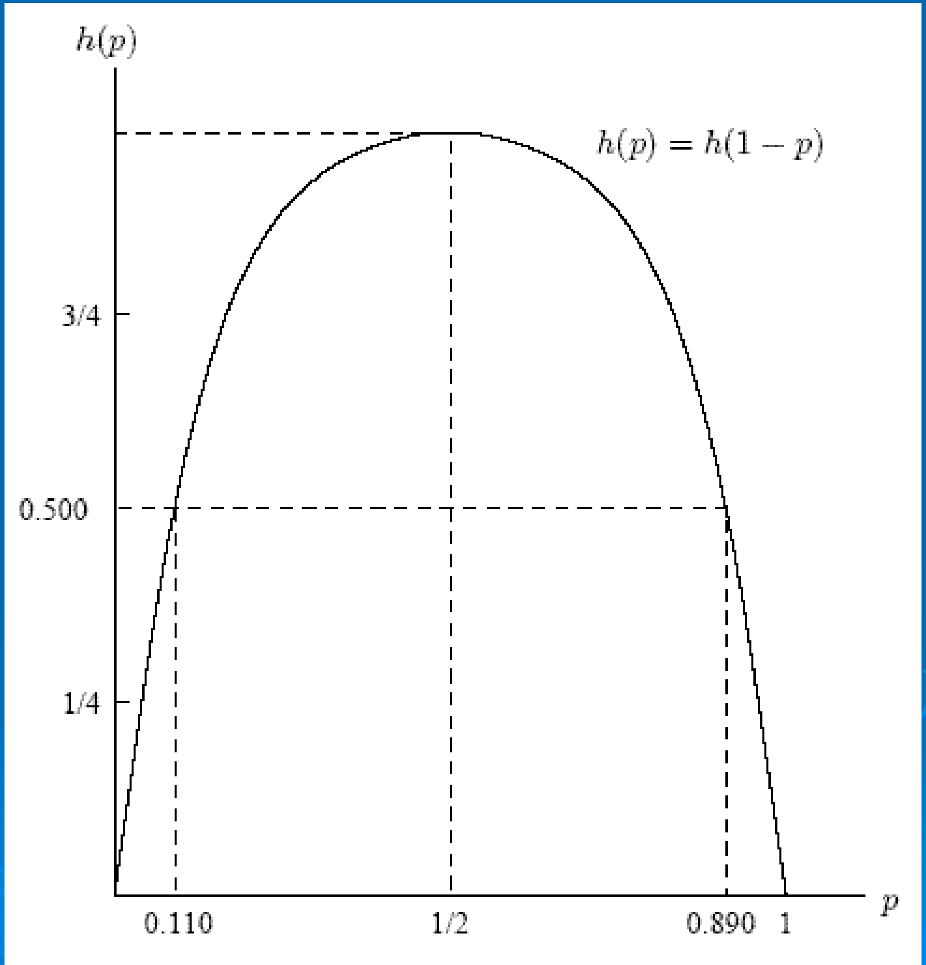


Figure 1.2: The Binary Entropy Function

Redundancy

- Using more bits for messages than their Entropy (absolute minimum) is called Redundancy.
  - Redundancy is the difference between message length in bits and the Entropy
  - Reducing redundancy gives us compression
  - This is how “winzip” or “gzip” work.

- A simple example.

This is excellent. See you later.	
Ths s XLnt. cu l8r	

Some redundancy removed.

- Redundancy can be used to correct errors.
- Less redundancy means less error tolerance.
- How much redundancy does the English language have?
  - Quite a lot
  - e.g. delete every 5th or so character of a news article. You will still be able to understand the content.

Entropy of the English Language

- What's the Entropy of the English language, i.e. how much information does it have per letter?
- 26 letters
- If all letters were equally likely (i.e. have the same frequency):
  - $H(X) = \log_2(26) = 4.7$ bits
- Some letters are more common than others.

Entropy of the English Language

English has the following distribution of letter frequency/probability:

e: 0.1241	m: 0.0281
t: 0.0969	f: 0.0235
a: 0.0820	p: 0.0203
i: 0.0768	y: 0.0189
n: 0.0764	g: 0.0181
o: 0.0714	w: 0.0135
s: 0.0706	v: 0.0124
r: 0.0668	b: 0.0106
l: 0.0448	k: 0.0039
d: 0.0363	x: 0.0021
h: 0.0350	j: 0.0019
c: 0.0344	q: 0.0009
u: 0.0287	z: 0.0005

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$
$$= 3.9\, bits\ /\ letter \qquad ("rate\ of\ \ English")$$

We can also look at combination of letters.
Two letters: Digrams
- For example: TH and EN are much more common than QZ

Three letters: Trigrams
N-letters: N-grams

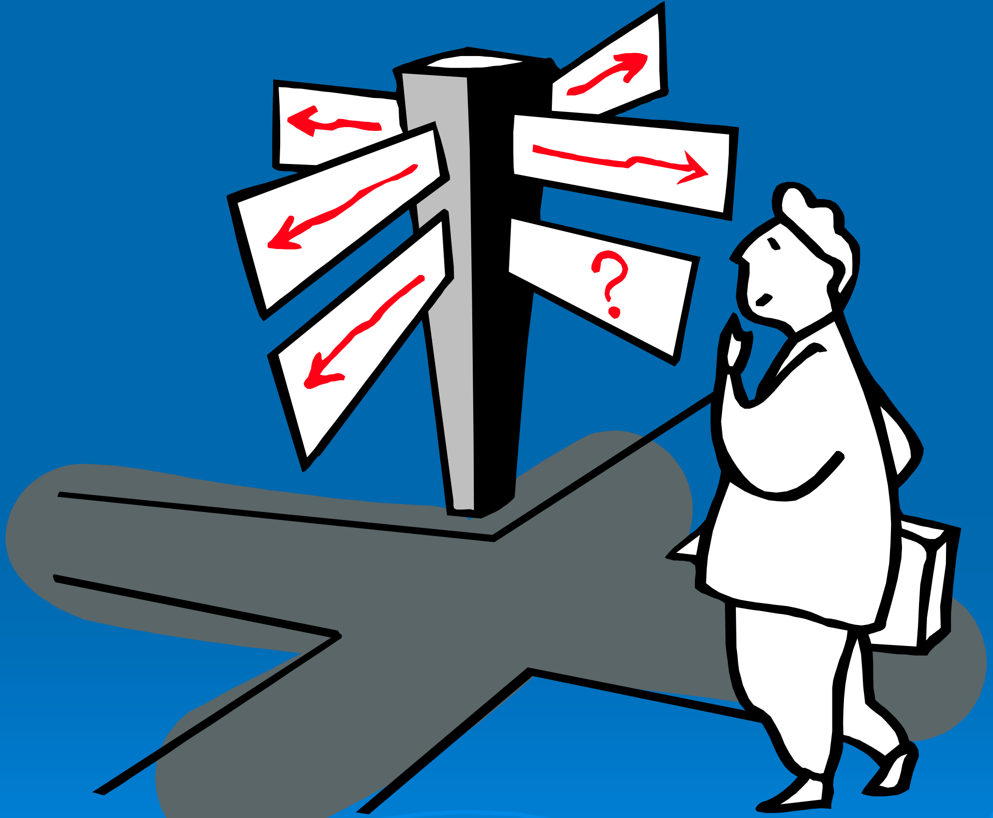
If we do the same kind of analysis for N-grams with increasingly large N, the entropy converges to a value of ≈1.5 bits/letter

Text encoded in ASCII uses 8 bits per letter. A very good compression Algorithm should be able to compress English text by a factor of ≈ 5

Password Entropy

- Entropy is often used as a quality measure of passwords
- High Entropy means more random, more unpredictable, lower level of ‘guessability’
- As Shannon has taught us, password Entropy does not only depend on the total number of possible passwords (N), but also their frequencies (probabilities)
  - See this paper for more details, in particular Section 3.1:
  - http://research.microsoft.com/pubs/227130/WhatsaSysadminToDo.pdf

Any questions so far?



Introduction to Cryptography

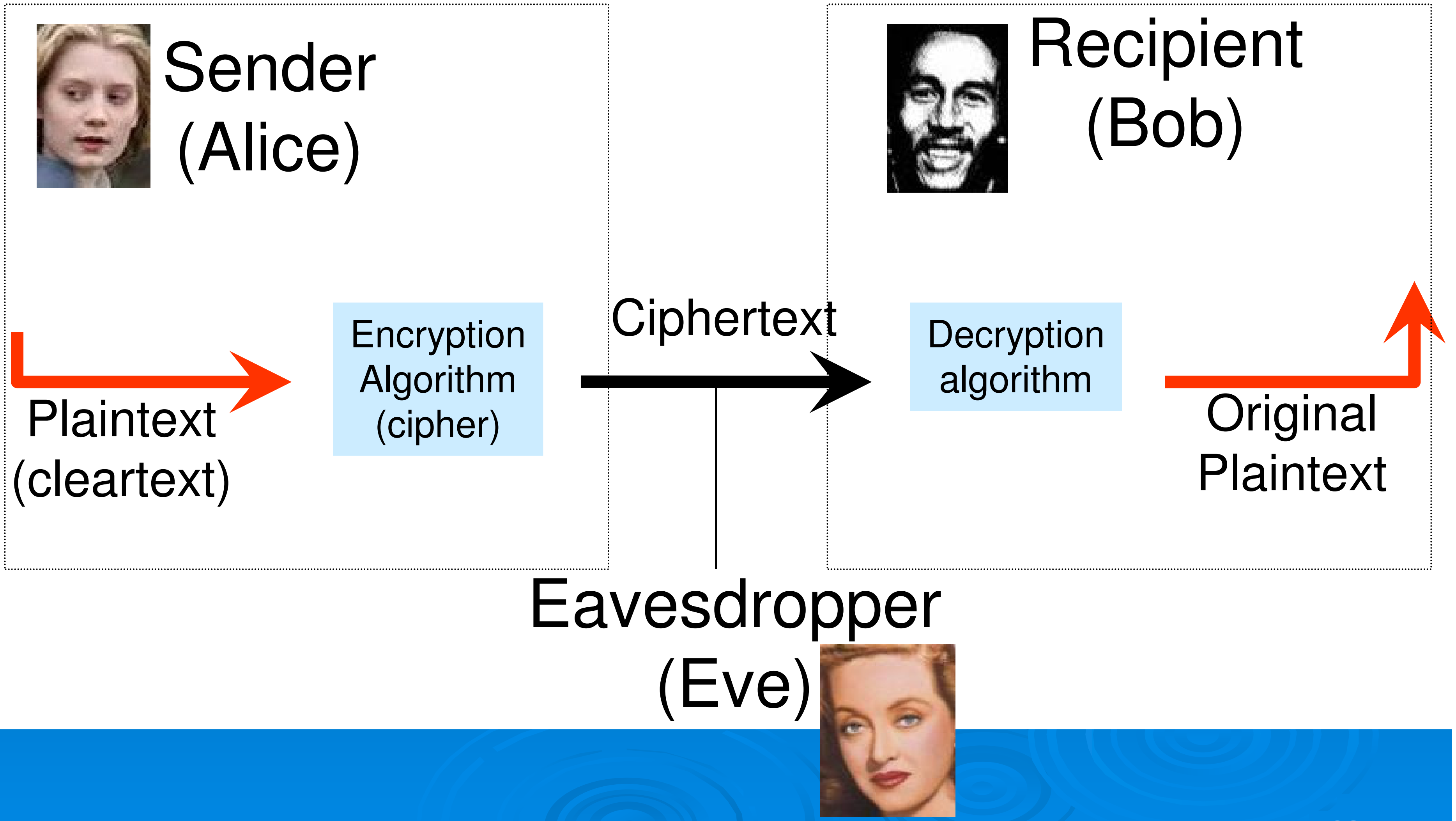
What is Cryptography?

- Cryptography
  - Literally “secret writing”
  - Science & art (practice) of keeping messages secure
- Cryptanalysis
  - Science & art (practice) of breaking message security
- Cryptology
  - “Secret words” Science of secret communications (theory and mathematics) associated with cryptography and cryptanalysis

Cryptography

- Primary aim:
  - **Confidentiality**: to allow a sender to transmit a message to a receiver, being sure that nobody else can read and understand the message.
- Secondary aims:
  - **Authenticity**: to prevent an attacker from impersonating the sender
  - **Integrity**: to convince the receiver that the message has not been altered in
  - **Non-repudiation**: the sender cannot later deny having sent the message.

Basic Model - Terminology



Cryptographic Algorithms, "Ciphers", "Code"

The central part of the basic model is the use of two algorithms: encryption and decryption.

The encryption of the plaintext *P* is the ciphertext *C*

$$C = E(P)$$

The decryption of the ciphertext *C* is the plaintext *P*

$$P = D(C)$$

In general, we have the identity:

$$P = D(\ E(P)\ )$$

Cryptographic Algorithms (Ciphers)

- We have to ensure that only legitimate or authorised users are able to (encrypt and) decrypt
- A simple way to do this is to restrict knowledge of the algorithms to authorised individuals.
  - **Keep Encryption/Decryption algorithm secret**
- **Problem ?**
  - This would mean that for every different set of people who wanted to communicate, there would have to be a different algorithm.
  - It becomes very impractical to keep using different algorithms – it is rather difficult to invent new ones which are resistant to cryptanalysis.
  - If an encryption algorithm gets in the hand of an attacker, we need to invent a new algorithm.
  - What is a better approach?

Cryptographic Keys

- Modern algorithms are actually large classes of encryption and decryption functions
- The sender and receiver need to use the corresponding functions
- The “index” of the function is called the **key**
  - The function is parameterised by the key