Lecture 16: Mandatory Access Control

CS 181S

November 7, 2018

Review: Access control

- Subject: principal to which execution can be attributed
- Object: data or resource
- Operation: performed by subject on object
- Right: entitlement to perform operation

Review: DAC

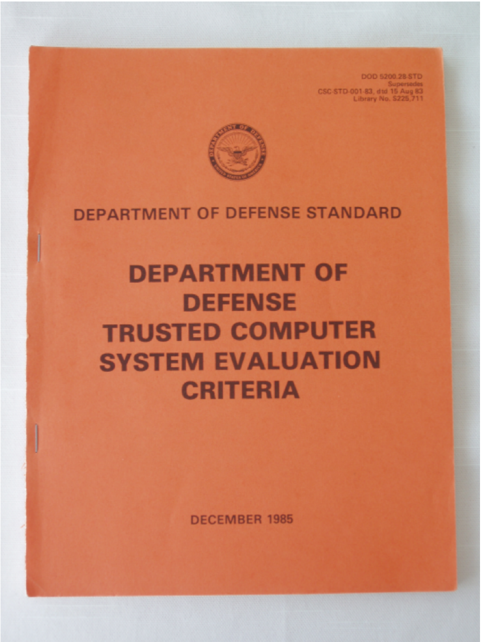
- Discretionary access control (DAC)
  - **Philosophy:** users have the *discretion* to specify policy themselves
  - Commonly, information belongs to the **owner** of object
  - Model: access control **relation**
    - Set of triples (subj,obj,rights)
    - Sometimes described as access control "matrix"
- Implementations:
  - Access control lists (ACLs): each object associated with list of (subject, rights)
  - Capability lists: each subject associated with list of (object, rights)
  - Capabilities: distributed ways of implementing privilege lists

MAC

- Mandatory access control (MAC)
  - not Message Authentication Code (applied crypto), nor Media Access Control (networking)
  - **philosophy:** central authority *mandates* policy
  - information belongs to the authority, not to the individual users

Multi-Level Security

- A mechanism for monitoring access control in a system where both principals and objects have security labels drawn from a hierarchy of labels
- Commonly associated with military systems
- Influenced "Orange Book" (DoD Trusted Computer System Evaluation Criteria)
  - A) Verified Protection
  - B) Mandatory Protection
  - C) Discretionary Protection
  - D) Minimal Protection



Sensitivity

- Concern is **confidentiality** of information
- Documents classified according to sensitivity: risk associated with release of information
- In US:
  - Top Secret
  - Secret
  - Confidential
  - Unclassified

Compartments

- Documents classified according to compartment(s): categories of information (in fact, aka category)
  - cryptography
  - nuclear
  - biological
  - reconnaissance
- **Need to Know Principle:** access should be granted only when necessary to perform assigned duties (instance of Least Privilege)
  - {crypto, nuclear}: must need to know about **both** to access
  - {}: no particular compartments

Labels

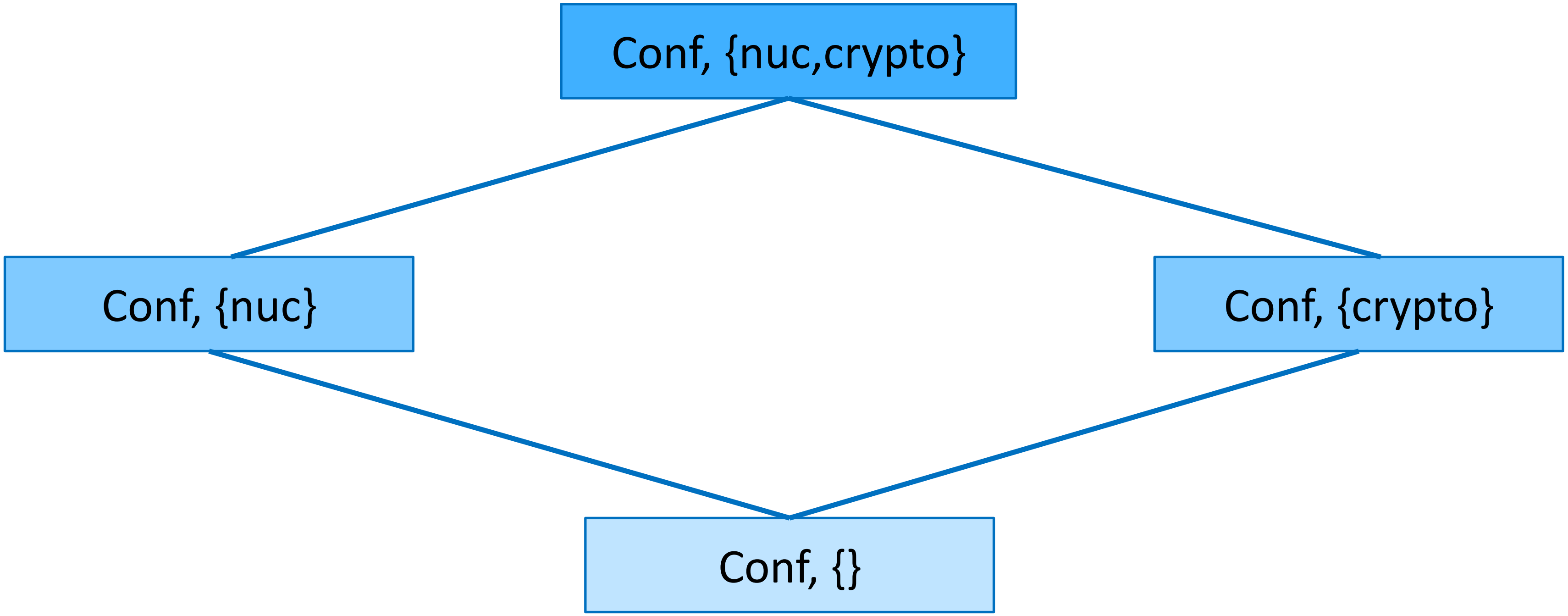
- Label: pair of sensitivity level and set of compartments, e.g.,
  - (Top Secret, {crypto, nuclear})
  - (Unclassified, {})
- Document is labeled aka classified
  - Perhaps each paragraph labeled
  - Label of document is most restrictive label for any paragraph
- Users are labeled according to their clearance
  - Users trustworthy by virtue of vetting process for security clearance
  - Out of scope (e.g.): user who views Top Secret information and calls the *Washington Post*
- Labels are imposed by organization
- **Notation:** let L(X) be the label of entity X

Restrictiveness of labels

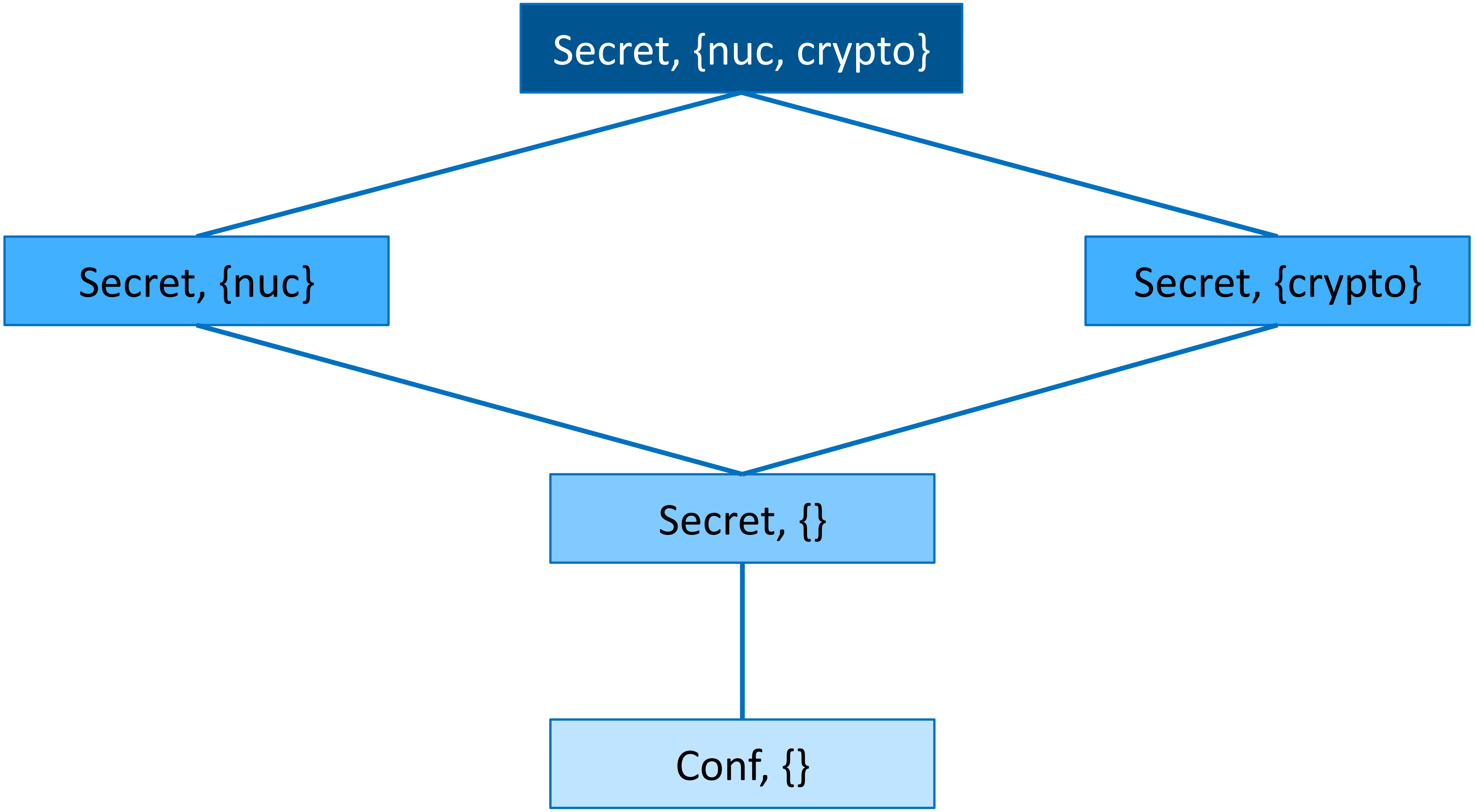
Notation: L1 ⊑ L2

- means L1 is no more restrictive than L2
  - less precisely: L1 is less restrictive than L2
- **Definition:**
  - Let L1 = (S1, C1) and L2 = (S2, C2)
  - L1 ⊑ L2 iff S1 ≤ S2 and C1 ⊆ C2
  - Where ≤ is order on sensitivity:
    Unclassified ≤ Confidential ≤ Secret ≤ Top Secret
- e.g.
  - (Unclassified,{}) ⊑ (Top Secret, {})
  - (Top Secret, {crypto}) ⊑ (Top Secret, {crypto,nuclear})

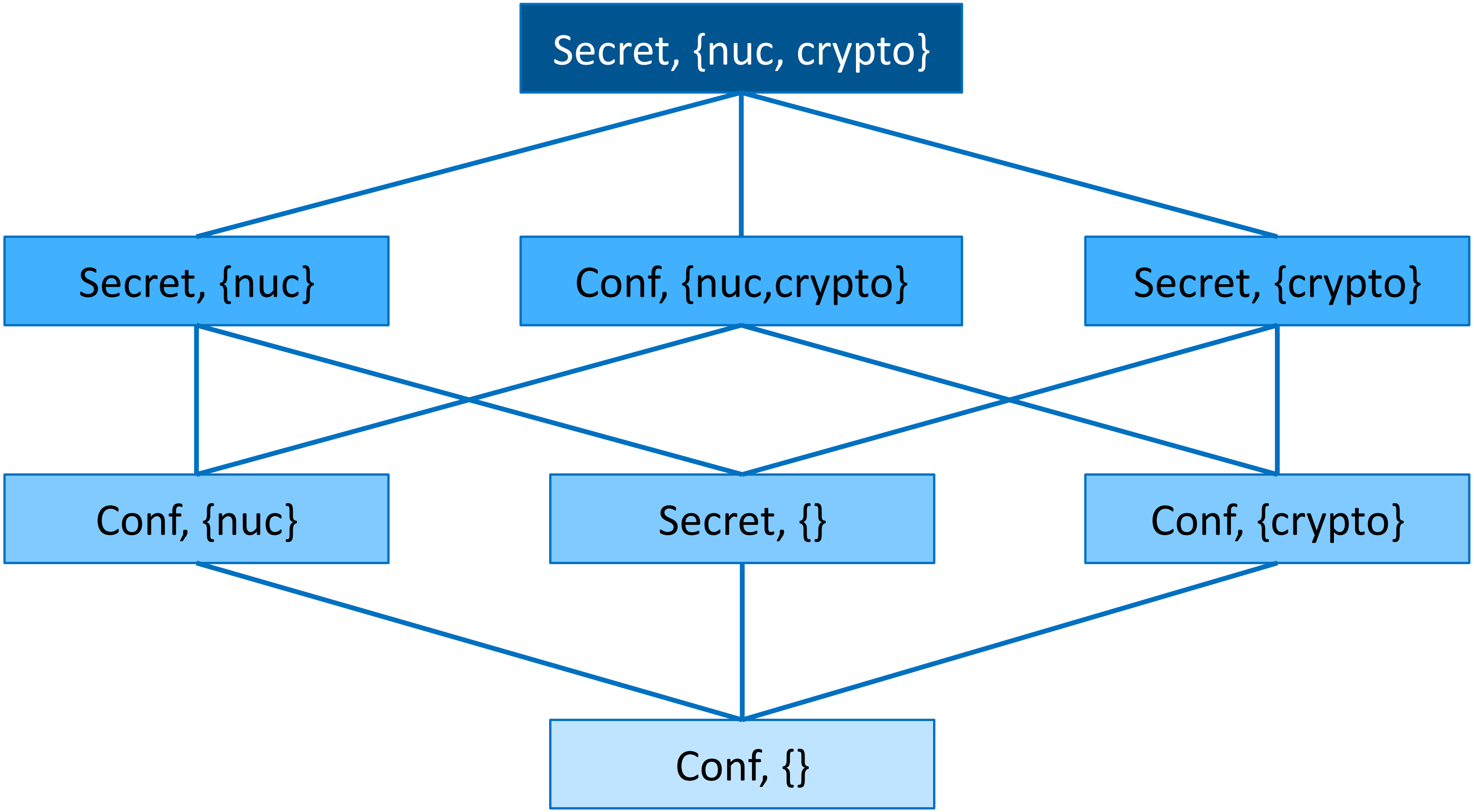
Label partial order



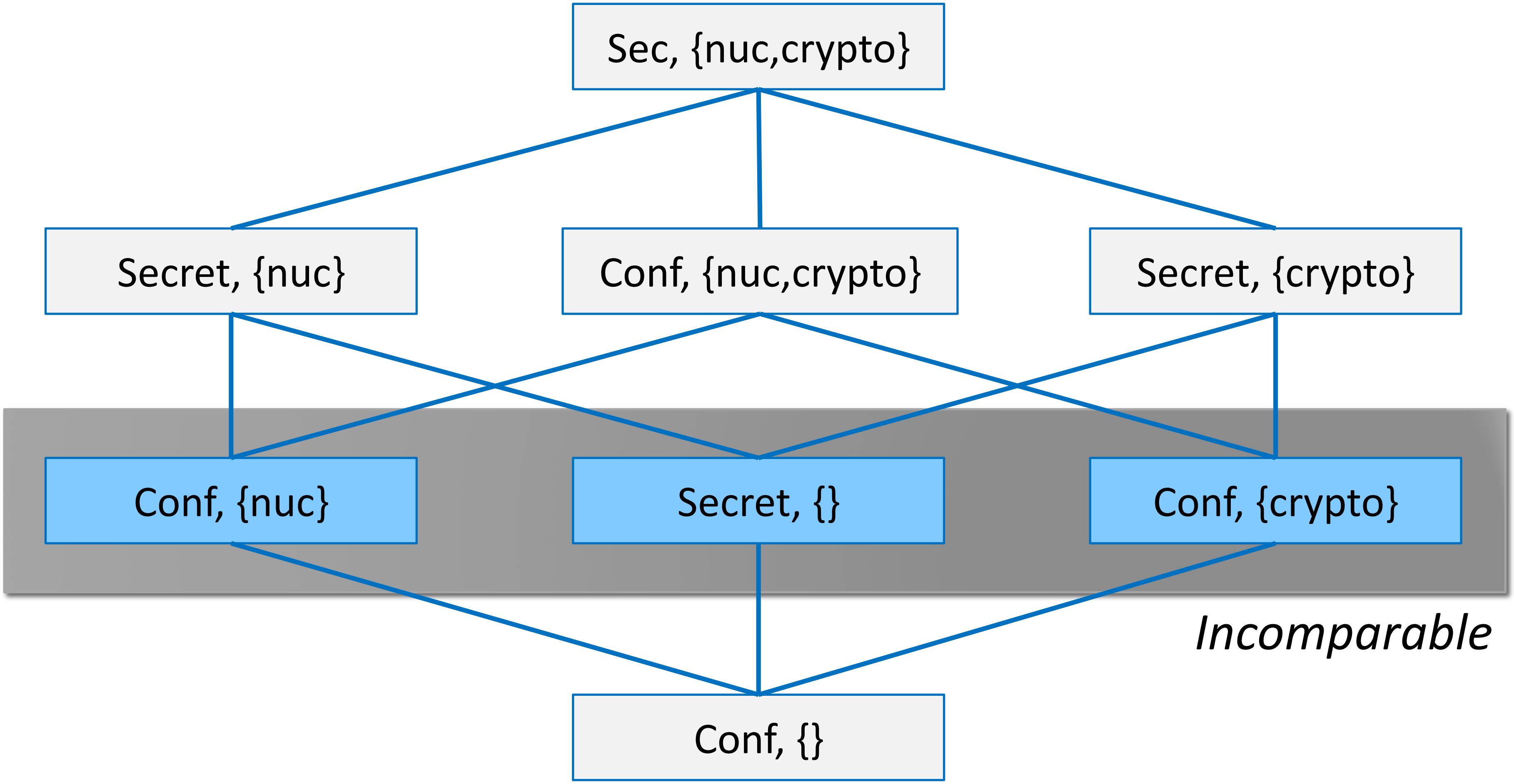
Label partial order



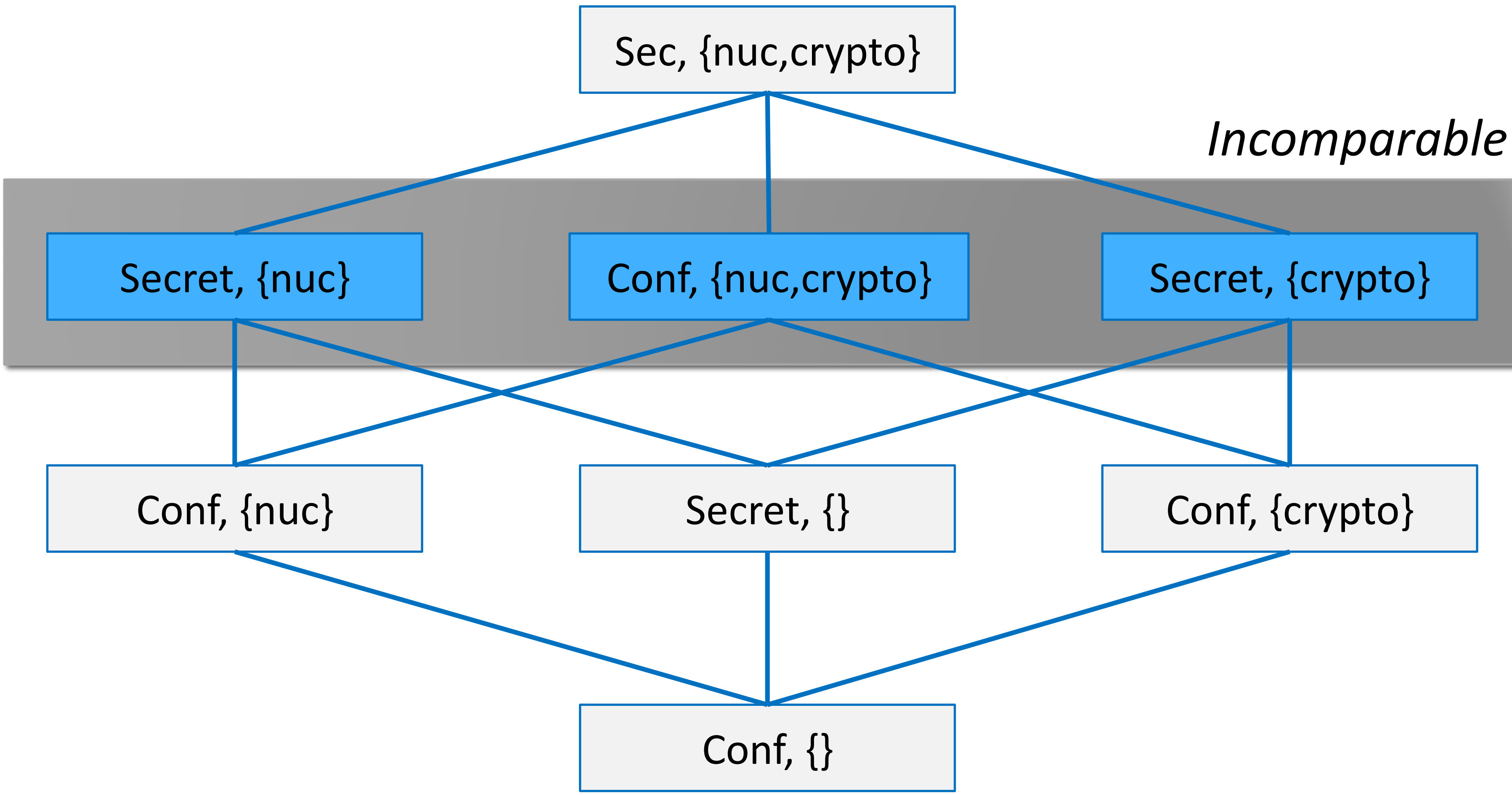
Label partial order



Label partial order



Label partial order



Access control with MLS

- When may a subject read an object?
  - **Threat:** subject attempts to read information for which it is not cleared
  - e.g., subject with clearance Unclassified attempts to read Top Secret information
- When may a subject write an object?
  - **Threat:** subject attempts to *launder* information by writing into a lower-security object
  - e.g., subject with clearance Top Secret reads Top Secret information then writes it into an Unclassified file

Access control with MLS

- When may a subject read an object?
  - **S may read O iff $L(O) \sqsubseteq L(S)$**
  - object's classification must be below (or equal to) subject's clearance
  - "no read up"
- When may a subject write an object?
  - **S may write O iff $L(S) \sqsubseteq L(O)$**
  - object's classification must be above (or equal to) subject's clearance
  - "no write down"

Reading with MLS

- Scenario:
  - Colonel with clearance (Secret, {nuclear, Europe})
  - DocA with classification (Confidential, {nuclear})
  - DocB with classification (Secret, {Europe, US})
  - DocC with classification (Top Secret, {nuclear, Europe})
- Which documents may Colonel **read**?
  - Recall: S may read O iff $L(O) \sqsubseteq L(S)$
  - DocA: (Confidential, {nuclear}) $\sqsubseteq$ (Secret, {nuclear, Europe})
  - DocB: (Secret, {Europe, US}) $\not\sqsubseteq$ (Secret, {nuclear, Europe})
  - DocC: (Top Secret, {nuclear, Europe}) $\not\sqsubseteq$ (Secret, {nuclear, Europe})

Writing with MLS

- Scenario:
  - Colonel with clearance (Secret, {nuclear, Europe})
  - DocA with classification (Confidential, {nuclear})
  - DocB with classification (Secret, {Europe, US})
  - DocC with classification (Top Secret, {nuclear, Europe})
- Which documents may Colonel **write**?
  - Recall: S may write O iff $L(S) \sqsubseteq L(O)$
  - DocA: (Secret, {nuclear, Europe}) $\not\sqsubseteq$ (Confidential, {nuclear})
  - DocB: (Secret, {nuclear, Europe}) $\not\sqsubseteq$ (Secret, {Europe, US})
  - DocC: (Secret, {nuclear, Europe}) $\sqsubseteq$ (Top Secret, {nuclear, Europe})

Reading and writing with MLS

- Scenario:
  - Colonel with clearance (Secret, {nuclear, Europe})
  - DocA with classification (Confidential, {nuclear})
  - DocB with classification (Secret, {Europe, US})
  - DocC with classification (Top Secret, {nuclear, Europe})
- Summary:
  - DocA: Colonel may read but not write
  - DocB: Colonel may neither read nor write
  - DocC: Colonel may write but not read

Prevention of laundering

- Earlier concern: "subject with clearance Top Secret reads Top Secret information then writes it into an Unclassified file"
- More generally:
  - S reads O1 then writes O2
  - where $L(O2) \sqsubset L(O1)$
  - and regardless of $L(S)$
- **Prohibited by MLS rules:**
  - S read O1, so $L(O1) \sqsubseteq L(S)$
  - S wrote O2, so $L(S) \sqsubseteq L(O2)$
  - So $L(O1) \sqsubseteq L(S) \sqsubseteq L(O2)$
  - Hence $L(O1) \sqsubseteq L(O2)$
  - But combined with $L(O2) \sqsubset L(O1)$, we have $L(O1) \sqsubset L(O1)$
  - Contradiction!
- So access control rules would defeat laundering, Trojan Horse, etc.

Perplexities of writing with MLS

1. Blind write: subject may not read higher-security object yet may write it
   - Useful for logging
   - Some implementations prohibit writing up as well as writing down
2. **User** who wants to write lower-security object may not
   - **Attenuation of privilege:** login at a lower security level than clearance
   - Motivated by Trojan Horse
   - Nice (annoying?) application of Least Privilege
3. Declassification violates "no write down"
   - Encryption or billing procedure produces (e.g.) Unclassified output from Secret information
   - Traditional solution is trusted subjects who are not constrained by access control rules

Formalizing MLS

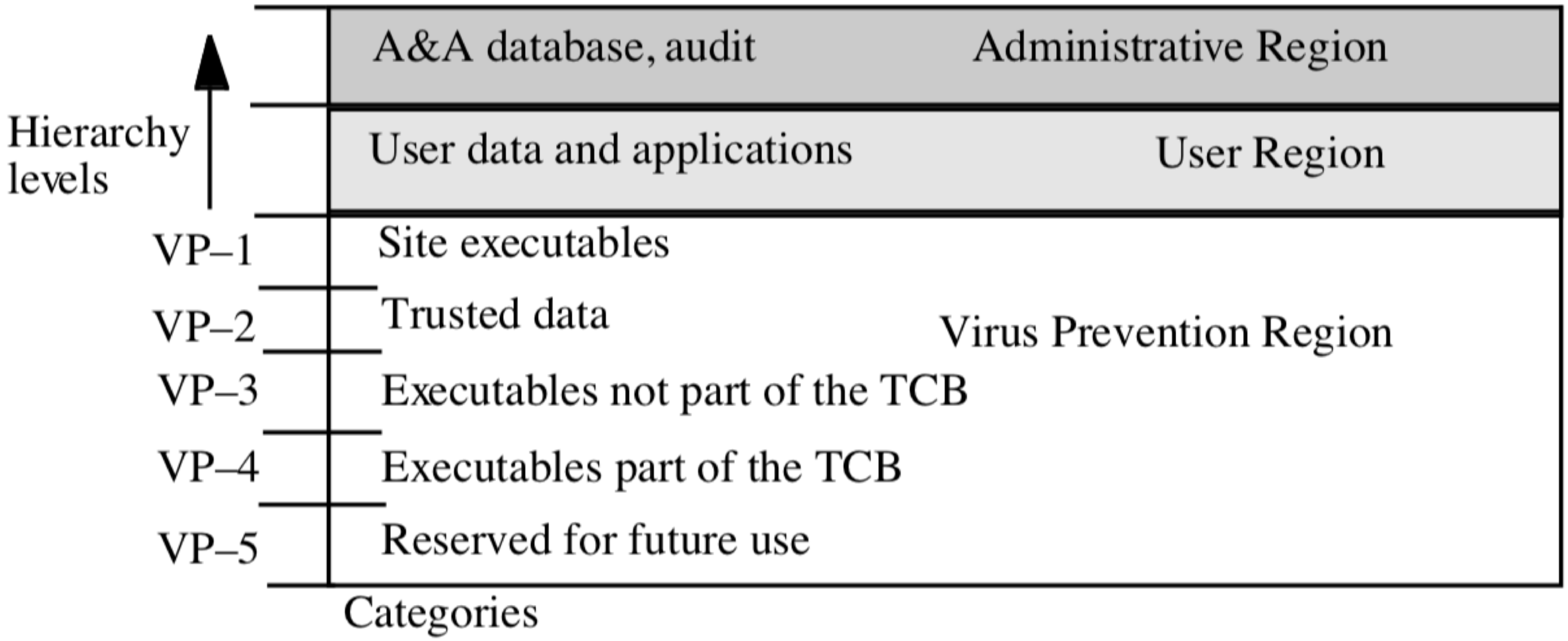
[Bell and LaPadula 1973]

- Formal mathematical model of MLS plus access control matrix
- Proof that information cannot leak to subjects not cleared for it
- "No read up": simple security property
- "No write down": *-property
- *"The influence of [BLP] permeates all policy modeling in computer security"* –Matt Bishop
  - Influenced Orange Book
  - Led to research field "foundations of computer security”

MLS in OSs

DG/UX

- Discontinued Unix OS, release 1985
- Three regions:
  Virus Protection $\sqsubseteq$ User Region $\sqsubseteq$ Administrative Region



MLS in OSs

DG/UX

- Discontinued Unix OS, release 1985
- Three regions:
  Virus Protection $\sqsubseteq$ User Region $\sqsubseteq$ Administrative Region
- MLS confidentiality: read down, no read up
- Extra integrity: no write down, no write up
  - for shared directories (e.g., /tmp), introduced mulit-level directories with one hidden subdirectory for each level

MLS in OSs

SELinux

- Kernel security module, dates back to NSA c. 2000, merged with Linux kernel mainline in 2.6
- Goal: separate security policy from security decisions
- Supports mandatory access controls in reference policy. When MLS is enabled:
  - Each principal (user or process) is assigned a context (username, role, domain, (sensitivity))
  - Each object (file, port, hardware) is assigned a context
  - SELinux enforces MLS

MLS in OSs

TrustedBSD [2000]

- Similar goals to SELinux: separate policy from security mechanism, implements MLS
- ported parts of SELinux to FreeBSD
- Many components eventually folded into FreeBSD
- Most interfaces supported on Macs since OSX 10.5

BLP, for integrity

- BLP is about confidentiality
- Adapted to integrity by Biba [1977]: same rules, different lattice
  - Instead of Unclassified and Secret, labels could be Untrusted and Trusted
- L1 $\sqsubseteq$ L2 means "L1 may flow to L2 without breaking confidentiality"
  - BLP: low secrecy sources may flow to high secrecy sinks
    - Hence Unclassified $\sqsubseteq$ Secret, but not v.v.
  - Biba: low integrity sources may not flow to high integrity sinks
    - Hence Trusted $\sqsubseteq$ Untrusted, but not v.v.
  - High vs. low is "flipped" (lattices are *duals*)

Biba model

- **S may read O iff $L(O) \sqsubseteq L(S)$**
  - E.g., Trusted subject cannot read Untrusted object
  - But Untrusted subject may read Trusted object
- **S may write O iff $L(S) \sqsubseteq L(O)$**
  - E.g., Trusted subject may write Untrusted object
  - But Untrusted subject may not write Trusted object

Beyond Multi-level Security…

Mandatory access control comes in many different forms (not just MLS):

1. Multi-level security (confidentiality, military)
2. Biba model (integrity, military)
3. Role-based access control (hybrid, organization)
4. Clark-Wilson (integrity, business)
5. Brewer-Nash (hybrid, consulting firm)

ROLE-BASED ACCESS CONTROL

Jobs

- Your access rights depend on job you are performing
  - Student in one class
  - TA in another class
  - Prof in another class?

- Existence of jobs is relatively stable in organization
  - Even if over time the people who perform them change jobs
  - Better not to directly assign rights to user

- Instead, associate rights with the job...

Roles and rights

Role: job function or title

- Users are assigned to roles
- Subjects executing on behalf of users can activate a role to indicate it is now performing that job
  - Least Privilege
  - Amplification of Privilege

Roles and rights

- Roles can be hierarchical
  - e.g. TA, prof
  - Hierarchy is a *partial order*
- Multiple roles may be active simultaneously
- Can be constraints on which roles users can simultaneously be assigned
  - e.g. cannot be both Student and TA in same course
  - provides possibility for Separation of Duty

Roles and rights

- **Rights:**
  - Rights are assigned to roles, not directly to users
  - Relation on (role, obj, rights)
- Role-based access control (RBAC) policy: role assignment plus rights assignment

Roles vs. groups

- Group:
  - set of users
  - can be assigned rights
- Role:
  - set of users
  - can be assigned rights
- **Differences?**
  - Roles are hierarchical and can inherit rights
  - Roles can be activated and deactivated

RBAC, DAC, MAC

Is RBAC a DAC or MAC policy?

- Role assignments typically dictated by organization: MAC
- Right assignments might come from organization or from owners of objects: MAC or DAC