February 25, 2014

- Hybrid models
  - Traducement
- Information flow
- Basics and background
  - Entropy
- Non-lattice flow policies

Case Study: Traducement

Designed to model electronic recordation

- What is recordation?
- Why do it electronically?
- Models and recordation
- Example: approach and problems

Recordation

- Recording title to real property
  - Real estate purchases
- Recording liens, *etc*.
  - Mortgage holders and such
- In California, County Recorders do this
  - No standards other than statutory ones
  - No state office oversees them

Goals of Recordation

- Establish title
- Establish priority of liens, *etc*.
- Protection of Public
  - Permanence of records
  - Fraud prevention (no secret conveyance, *etc*.)
- Recording triggers release of funds
  - It's the official record of property ownership

Requirements of a Solution

1. A signed document cannot be altered (although new signatures may be appended);
2. A document may require multiple signatures;
3. A document submitted to the recorder's office may be revoked by any signatory until the document is recorded, but is no longer eligible for additional signatures;
4. The recorder may only append information to the document (*i.e.*, sign it); and
5. If the document is recorded, it becomes a public record immutable to all parties.

How to Record Something

- Submission
  - Presentation of documents to recorder
- Validation
  - Check for conformance with statutory requirements
  - Calculate fees
- Storage
  - Record documents, index and provide locators
  - Filming and/or imaging the documents to create archival record
- Return documents

Modeling the Process

- Confidentiality not an issue
  - Exception: some fees may be
- Integrity a *critical* issue
  - Originator must be able to file document
  - Document must be correct, legal
  - Document immutable
- Availability may, may not be issue

Electronic Commerce

- Model many are trying to use, but there are substantial differences:
  - Emphasis on privacy inappropriate
  - Nothing exchanged (no non-fungible property involved)
  - Not immutable; you can erase an electronic transaction
  - Does not establish title
  - Does not deal with liens

Traducement

- Model designed for electronic recordation
  - a signed document cannot be altered (although new signatures may be appended)
  - a document may require multiple signatures
  - a document submitted to the recorder's office may be revoked by any signatory until the document is recorded, but additional signatures may not be added
  - the recorder may only append information to the document (i.e., sign it)
  - if the document is recorded, it becomes a public record immutable to all parties.

Key Notions

- *Publishing* document
  - Cannot modify it further
  - Making it available to larger community
- *Signing* document
  - Associates authors with documents
- Common to legal documents
  - Unusual in other documents

Entities

- Subjects
  - *Authors* contribute in some way to the document to be filed
  - *Recorders* attest to the completion of document, converting it into official record
- Objects
  - Documents to be filed

Definitions

- Author set *AS*
  - Attribute of object that specifies set of users who wrote to object
  - No author can be removed from author set
- Signer set *SS*
  - Attribute that specifies users who approve the object, contents
  - Any reader can add themselves to this set

Create Rule

- User u creates object o:
  - o indelibly stamped with creation time
  - $o'(AS) = \{ u \}$
  - $o'(SS) = \varnothing$

Alteration Rule

- User u alters object o:
  - $o'(AS) = \{ u \} \cup o(AS)$
  - $o'(SS) = \varnothing$

Signature Rule

- User u signs object o:
  - $o'(AS) = o(AS)$
  - $o'(SS) = \{ u \} \cup o(SS)$

Example

- Peter drafts document
  - $d(AS) = \{ \text{Peter} \}$, $d(SS) = \varnothing$
- Paul approves
  - $d(AS) = \{ \text{Peter} \}$, $d(SS) = \{ \text{Paul} \}$
- Mary makes some changes
  - $d(AS) = \{ \text{Peter, Mary} \}$, $d(SS) = \varnothing$
- Everyone says it's fine
  - $d(AS) = \{ \text{Peter, Mary} \}$
  - $d(SS) = \{ \text{Peter, Paul, Mary} \}$

Copy Rule

- User u copies object o to O:
  - $O'(AS) = o(AS)$
  - $O'(SS) = o(SS)$

Proposition

- A user is in the *signer set* of an object if and only if the document has not been modified since the user was added to the signer set.
- *Proof*

  (⇒) Let $u \in o(SS)$. Creation, alteration rules set $o(SS) = \varnothing$; by induction, not used. Signature, copy do not alter $o(SS)$.

Proof (*con't*)

- *Proof*

  ($\Leftarrow$) Assume *o* not modified since *u* added to *o*(*SS*).

  - Signature or copy rule applied
  - Signature rule adds to *o*(*SS*); does not delete any elements
  - Copy rule copies original *o*(*SS*); does not delete any elements
  - Induction gives the result

Preconditions

1. Each document in the system has an author set list identifying all users who created or modified that document
2. Each document in the system has a signer set list identifying all users who approve that document.

Theorem

- If a system satisfies the preconditions, then the system still satisfies the preconditions after any sequence of applications of the creation, alteration, signature, and copy rules.
- *Proof*: Let a system satisfy preconditions in state s_0. Apply one of the rules to transition to state s_1.

Applying Rules

- Create rule
  - New document created; $o(AS)$ is creator only (#1 met) and $o(SS)$ empty (#2 met)
- Alteration rule
  - Add user to $o(AS)$, so $o(AS)$ contains only new user, members of old $o(AS)$ (#1 met); $o(SS)$ cleared, so no-one has approved of it (#2 met)

Applying Rules

- Signature rule
  - Document not changed so $o(AS)$ not changed (#1 met); add signer to $o(SS)$, as signer approves of (unchanged) document (#2 met)
- Copy rule
  - Create new instance of document, so no changes (#1 met); signers approved of content and no changes to that (#2 met)

Basic Security Theorem

- Analogue to Bell-LaPadula BST
- Define *secure*:
  - System meeting preconditions is secure
- Idea of theorem:
  - Begin in secure state
  - Apply transitions (rules)
  - Resulting system in secure state

Theorem

Let R be a rule, s be a state of a system, and s' be the state obtained by applying R to s. Let the system in state s satisfy Preconditions 1 and 2, and let O and O' be the set of objects in states s and s', respectively. Then:

1. If there is an object o' such that
   a) $o' \notin O$
   b) $o' \in O'$
   c) $O' = O \cup \{o'\}$
   d) $o'(AS) = \{u\}$ for some subject u
   e) $o'(SS) = \varnothing$

   then s' satisfies Preconditions 1 and 2.

Theorem

2. If there is an object $o \in O$ such that
   a) $o'(AS) = \{u\} \cup o(AS)$ for some subject u
   b) $o'(SS) = \varnothing$

   then s' satisfies Preconditions 1 and 2.
3. If there is an object $o \in O$ such that
   a) $o'(AS) = o(AS)$
   b) $o'(SS) = \{u\} \cup o(SS)$ for some subject u

   then s' satisfies Preconditions 1 and 2.

Theorem

4. If there is an object $x' \in O'$ such that:
   a) $x' \notin O$
   b) there is an object $o \in O$
   c) $x'(AS) = o(AS)$
   d) $x'(SS) = o(SS)$

   then s' satisfies Preconditions 1 and 2.

Proof (First Case Only)

- *s* satisfies Preconditions 1 and 2
- For each $o \in O$, $o(AS)$ identifies all users who created or modified *o*
- For each $o \in O$, $o(SS)$ identifies all users who approve *o*
- $o' \notin O$ but $o' \in O' \Rightarrow o'$ created
  - Let *u* be the creator

Proof (*con't*)

- $o'(AS) = \{u\}$
  - $o'(AS)$ contains user who created o'
- $o'(AS)$ identifies all users who created, modified o', satisfying precondition 1
- $o'(SS) = \varnothing$
  - o' just created, so no-one yet approves its contents
- $o'(SS)$ identifies all users who approved it, satisfying precondition 2

Naming

- How do you identify authors, signers?
  - Important as if two have the same name, you lose accountability
- Leads to *domain rule*: the authors contained in the author group shall be given unique names
  - Problem is understood, lots of approaches to solving it (X.509 certificate hierarchies, etc.)
  - Call these *fully qualified names* (*FQN*)

Authorship Integrity

- Definition of terms
  - *domain* collection of systems
  - *subdomain* an inferior domain
  - *parent domain* a superior domain

  Each domain has its own administrative authority

Note: theorems hold as long as signers use FQNs

Goal: Record Information

An object o is *recorded* when

1. $o(AS) \subseteq o(SS)$; and
2. the recorder's office executes a recordation transformation on the object.

Designated repository: stores a copy of every recorded object in its domain.

Review Requirements

1. A signed document cannot be altered (although new signatures may be appended);
   - See alteration rule
2. A document may require multiple signatures;
   - See signature rule
3. A document submitted to the recorder's office may be revoked by any signatory until the document is recorded, but is no longer eligible for additional signatures;
   - See alteration rule
   - Definition of recorder's transformation

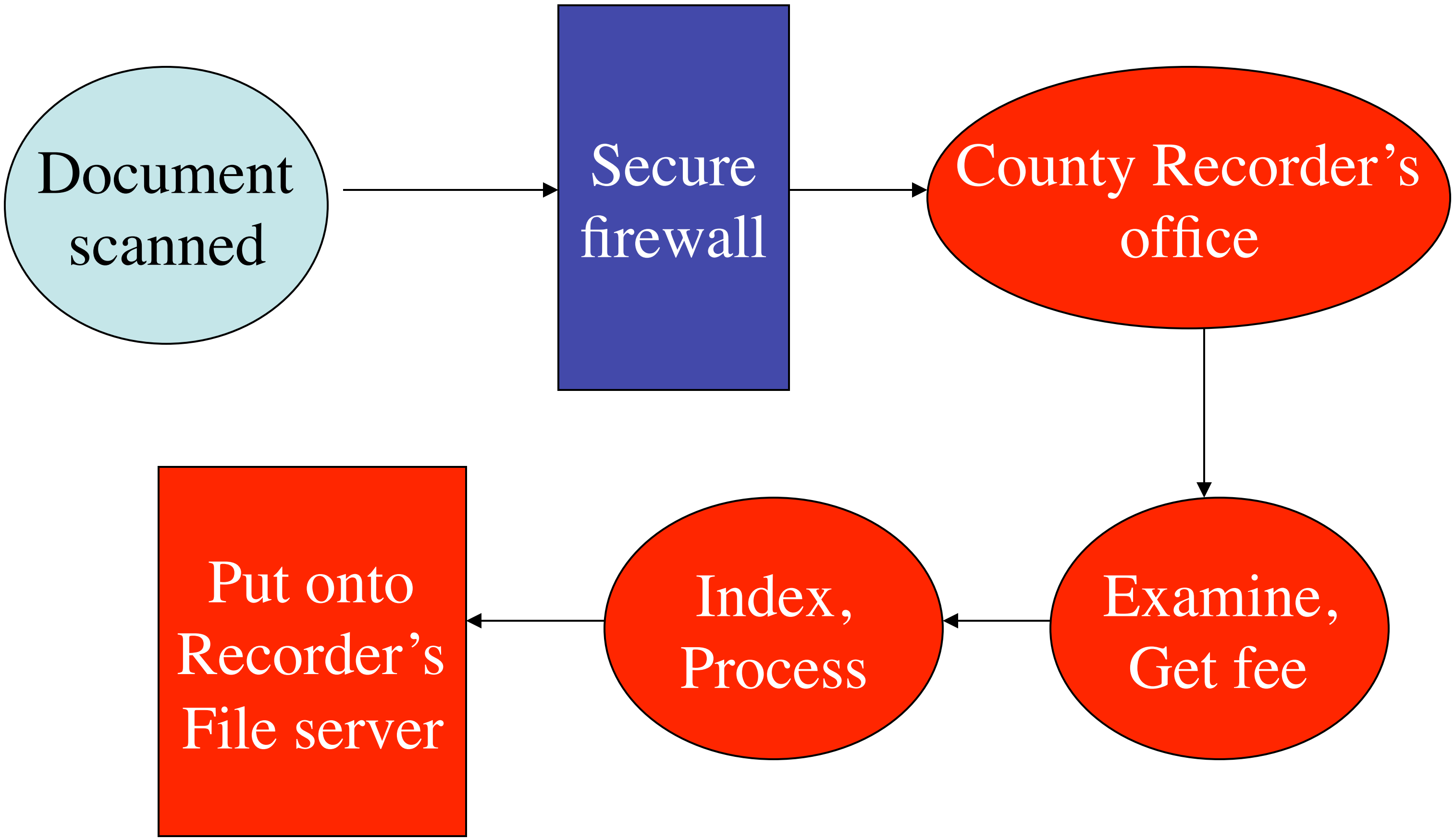
Review Requirements

4. The recorder may only append information to the document (*i.e.*, sign it); and
5. If the document is recorded, it becomes a public record immutable to all parties.
   – Definition of *recorder's transformation*

Now What?

- Can identify characteristics of a solution
  - If designing a solution, it must have those characteristics
- Know what to look for on a claimed solution

Basic Approach In Use



Assumptions

- Trusted relationship between author of images and recording authority
  - Encryption, acknowledgements
  - NB: Acknowledgement is “standard form wherein the author of the image acknowledges in writing that the documents submitted have original seals and signatures”

Submission of Documents

- How do you know the document received was the same as the one intended to be recorded?
  - Threat: I change the document in transit, before, or after it was sent
  - Digital signature assures document unchanged since signed and binds document to a public key
  - Public key infrastructure (PKI) binds public keys to principles (users)

Questions

- Is the user signing lawfully authorized to sign?
  – Albert di Salvo gets a real estate license …
- Is the user requesting the signature the one authorized to request the signature?
  – Sharing passwords, sharing a system … spoofing
- Is document changed between the user requesting the signature and the document being signed?
  – Virus-like programs change it first (use Adobe Photoshop-like program to change stamps, for example), unbeknownst to the user

More Questions

- Is the right public key used to sign the document?
  - PKI assumes certificates, binding keys to users, are issued to the right people
- Did the submitter change the document without the other party's consent?
  - On paper, this can usually be detected
  - Electronically, no way, unless original document digitally signed (see above)

Validation and Storage

- Document arrives at server
  - Stored in one area; validated here
  - When recorded, moved to permanent area
    - Burned onto CD or some other WORM media
- Operating system, web servers, other supporting applications provide security

Questions

- What is the system connected to?
  - Where can attackers come from?
- How well will the operating system withstand penetration attempts?
  - Lots of vulnerabilities in all software, OSes
- What operational security procedures are in place to maintain the security?
  - Bad procedures can weaken the best system
  - Who installs security patches, keeps up to date with new attacks, holes?

More Questions

- Is digital signature stored with document?
  - On the validation server
    - If not, it can be changed there
  - On the archive server
    - If not, no way to revalidate that document was same as sent

Return Documents

(Read this as retrieval of documents)

- Someone requests a title or copies of liens
  - Retrieval system gets it and presents it

Questions

- How do you know it gets the right one?

  Example: three documents about your house
  - The first (real) one says you have paid off all liens on your house.
  - The second (bogus) one puts a lien on your house.
  - The third (bogus) one forecloses on your house.
  - Which one is returned?

Solving the Problem

- AB 578 directs CA Attorney General to establish standards for electronic recordation systems
  - Includes security testing
- National efforts under way, too

The Problem With Solutions

- Vendor: “This system is designed and built using standard industrial software engineering techniques”
- Customer: “We installed and run this following the vendor’s instructions”

- Took 5 minutes to gain illicit, unauthorized access to system
- Took 10 minutes to compromise system’s functioning so it reported incorrect results
- Took 20 minutes to find all “hidden” passwords embedded in programs

Moral: current software and systems are not secure!

Information Flow

- How do we define and measure it?
  - *Entropy*
- So, let's review entropy

Entropy

- Uncertainty of a value, as measured in bits
- Example: X value of fair coin toss; X could be heads or tails, so 1 bit of uncertainty
  - Therefore entropy of X is $H(X) = 1$
- Formal definition: random variable X, values $x_1, \ldots, x_n$; so $\Sigma_i \, \mathrm{p}(X = x_i) = 1$

  $H(X) = -\Sigma_i \, p(X = x_i) \lg p(X = x_i)$

Heads or Tails?

- $H(X) = -p(X = \text{heads}) \lg p(X = \text{heads})$
  $-\text{p}(X = \text{tails}) \lg p(X = \text{tails})$
  $= -(1/2) \lg (1/2) - (1/2) \lg (1/2)$
  $= -(1/2)(-1) - (1/2)(-1) = 1$
- Confirms previous intuitive result

n-Sided Fair Die

$H(X) = -\Sigma_i\, p(X = x_i) \lg p(X = x_i)$

As $p(X = x_i) = 1/n$, this becomes

$H(X) = -\Sigma_i\, (1/n) \lg (1/ n) = -n(1/n)\, (-\lg n)$

so

$H(X) = \lg n$

which is the number of bits in n, as expected

Ann, Pam, and Paul

Ann, Pam twice as likely to win as Paul

W represents the winner. What is its entropy?

- w_1 = Ann, w_2 = Pam, w_3 = Paul
- $p(W = w_1) = p(W = w_2) = 2/5$, $p(W = w_3) = 1/5$

- So $H(W) = -\Sigma_i \, p(W = w_i) \lg p(W = w_i)$

  $= -(2/5) \lg (2/5) - (2/5) \lg (2/5) - (1/5) \lg (1/5)$

  $= -(4/5) + \lg 5 \approx 1.52$

- If all equally likely to win, $H(W) = \lg 3 = 1.58$

Joint Entropy

- X takes values from $\{ x_1, \ldots, x_n \}$
  - $\Sigma_i \, p(X = x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$
  - $\Sigma_i \, p(Y = y_i) = 1$
- Joint entropy of X, Y is:
  - $H(X, Y) = -\Sigma_j \, \Sigma_i \, p(X=x_i, Y=y_j) \lg p(X=x_i, Y=y_j)$

Example

X: roll of fair die, Y: flip of coin

$p(X{=}1, Y{=}\text{heads}) = p(X{=}1)\ p(Y{=}\text{heads}) = 1/12$

- As X and Y are independent

$H(X, Y) = -\Sigma_j\ \Sigma_i\ p(X{=}x_i, Y{=}y_j) \lg p(X{=}x_i, Y{=}y_j)$

$= -2\ [\ 6\ [\ (1/12) \lg (1/12)\ ]\ ] = \lg 12$

Conditional Entropy

- X takes values from $\{ x_1, \ldots, x_n \}$
  - $\Sigma_i \, p(X=x_i) = 1$
- Y takes values from $\{ y_1, \ldots, y_m \}$
  - $\Sigma_i \, p(Y=y_i) = 1$
- Conditional entropy of X given $Y=y_j$ is:
  - $H(X \mid Y=y_j) = -\Sigma_i \, p(X=x_i \mid Y=y_j) \lg p(X=x_i \mid Y=y_j)$
- Conditional entropy of X given Y is:
  - $H(X \mid Y) = -\Sigma_j \, p(Y=y_j) \, \Sigma_i \, p(X=x_i \mid Y=y_j) \lg p(X=x_i \mid Y=y_j)$

Example

- X roll of red die, Y sum of red, blue roll
- Note $p(X=1 \mid Y=2) = 1$, $p(X=i \mid Y=2) = 0$ for $i \neq 1$
  - If the sum of the rolls is 2, both dice were 1
- $H(X|Y=2) = -\Sigma_i\, p(X=x_i \mid Y=2) \lg p(X=x_i \mid Y=2) = 0$
- Note $p(X=i\ , Y=7) = 1/6$
  - If the sum of the rolls is 7, the red die can be any of 1, …, 6 and the blue die must be 7–roll of red die
- $H(X|Y=7) = -\Sigma_i\, p(X=x_i \mid Y=7) \lg p(X=x_i \mid Y=7)$

$$= -6\ (1/6) \lg (1/6) = \lg 6$$

Perfect Secrecy

- Cryptography: knowing the ciphertext does not decrease the uncertainty of the plaintext
- $M = \{ m_1, \ldots, m_n \}$ set of messages
- $C = \{ c_1, \ldots, c_n \}$ set of corresponding ciphertext
- Cipher $c_i = E(m_i)$ achieves *perfect secrecy* if $H(M \mid C) = H(M)$

Entropy and Information Flow

- Idea: info flows from *x* to *y* as a result of a sequence of commands *c* if you can deduce information about *x* before *c* from the value in *y* after *c*
- Formally:
  - *s* time before execution of *c*, *t* time after
  - $H(x_s \mid y_t) < H(x_s \mid y_s)$
  - If no *y* at time *s*, then $H(x_s \mid y_t) < H(x_s)$

Example 1

- Command is $x := y + z$; where:
  - $0 \le y \le 7$, equal probability
  - $z = 1$ with prob. 1/2, $z = 2$ or 3 with prob. 1/4 each
- s state before command executed; t, after; so
  - $H(y_s) = H(y_t) = -8(1/8) \lg (1/8) = 3$
  - $H(z_s) = H(z_t) = -(1/2) \lg (1/2) - 2(1/4) \lg (1/4) = 1.5$
- If you know x_t, y_s can have at most 3 values, so $H(y_s \mid x_t) = -3(1/3) \lg (1/3) = \lg 3$

Example 2

- Command is
  - **if** $x = 1$ **then** $y := 0$ **else** $y := 1$;

  where:
  - x, y equally likely to be either 0 or 1
- $H(x_s) = 1$ as x can be either 0 or 1 with equal probability
- $H(x_s \mid y_t) = 0$ as if $y_t = 1$ then $x_s = 0$ and vice versa
  - Thus, $H(x_s \mid y_t) = 0 < 1 = H(x_s)$
- So information flowed from x to y

Implicit Flow of Information

- Information flows from x to y without an *explicit* assignment of the form $y := f(x)$
  - $f(x)$ an arithmetic expression with variable x
- Example from previous slide:
  - **if** $x = 1$ **then** $y := 0$
    else $y := 1$;
- So must look for implicit flows of information to analyze program

Notation

- $\underline{x}$ means class of x
  - In Bell-LaPadula based system, same as "label of security compartment to which x belongs"
- $\underline{x} \le \underline{y}$ means "information can flow from an element in class of x to an element in class of y"
  - Or, "information with a label placing it in class $\underline{x}$ can flow into class $\underline{y}$"

Information Flow Policies

Information flow policies are usually:

- reflexive
  - So information can flow freely among members of a single class
- transitive
  - So if information can flow from class 1 to class 2, and from class 2 to class 3, then information can flow from class 1 to class 3

Non-Transitive Policies

- Betty is a confident of Anne
- Cathy is a confident of Betty
  - With transitivity, information flows from Anne to Betty to Cathy
- Anne confides to Betty she is having an affair with Cathy's spouse
  - Transitivity undesirable in this case, probably

Transitive Non-Lattice Policies

- 2 faculty members co-PIs on a grant
  - Equal authority; neither can overrule the other
- Grad students report to faculty members
- Undergrads report to grad students
- Information flow relation is:
  - Reflexive and transitive
- But some elements (people) have no "least upper bound" element
  - What is it for the faculty members?

Confidentiality Policy Model

- Lattice model fails in previous 2 cases
- Generalize: policy $I = (SC_I, \leq_I, join_I)$:
  - SC_I set of security classes
  - $\leq_I$ ordering relation on elements of SC_I
  - $join_I$ function to combine two elements of SC_I
- Example: Bell-LaPadula Model
  - SC_I set of security compartments
  - $\leq_I$ ordering relation *dom*
  - $join_I$ function *lub*

Confinement Flow Model

- $(I, O, confine, \rightarrow)$
  - $I = (SC_I, \leq_I, join_I)$
  - O set of entities
  - $\rightarrow$: $O \times O$ with $(a, b) \in \rightarrow$ (written $a \rightarrow b$) iff information can flow from a to b
  - for $a \in O$, $confine(a) = (a_L, a_U) \in SC_I \times SC_I$ with $a_L \leq_I a_U$
    - Interpretation: for $a \in O$, if $x \leq_I a_U$, info can flow from x to a, and if $a_L \leq_I x$, info can flow from a to x
    - So a_L lowest classification of info allowed to flow out of a, and a_U highest classification of info allowed to flow into a

Assumptions, *etc*.

- Assumes: object can change security classes
  - So, variable can take on security class of its data
- Object x has security class $\underline{x}$ currently
- Note transitivity *not* required
- If information can flow from a to b, then b dominates a under ordering of policy I:

$$(\forall\, a, b \in O)[\; a \rightarrow b \Rightarrow a_L \leq_I b_U \;]$$

Example 1

- SC_I = { U, C, S, TS }, with U $\leq_I$ C, C $\leq_I$ S, and S $\leq_I$ TS
- $a, b, c \in O$
  - confine(a) = [C, C]
  - confine(b) = [S, S]
  - confine(c) = [TS, TS]
- Secure information flows: $a \rightarrow b$, $a \rightarrow c$, $b \rightarrow c$
  - As $a_L \leq_I b_U$, $a_L \leq_I c_U$, $b_L \leq_I c_U$
  - Transitivity holds

Example 2

- SC_I, $\leq_I$ as in Example 1
- $x, y, z \in O$
  - confine(x) = [C, C]
  - confine(y) = [S, S]
  - confine(z) = [C, TS]
- Secure information flows: $x \rightarrow y$, $x \rightarrow z$, $y \rightarrow z$, $z \rightarrow x$, $z \rightarrow y$
  - As $x_L \leq_I y_U$, $x_L \leq_I z_U$, $y_L \leq_I z_U$, $z_L \leq_I x_U$, $z_L \leq_I y_U$
  - Transitivity does not hold
    - $y \rightarrow z$ and $z \rightarrow x$, but $y \rightarrow x$ is false, because $y_L \leq_I x_U$ is false

Transitive Non-Lattice Policies

- Q = (S_Q, $\leq_Q$) is a *quasi-ordered set* when $\leq_Q$ is transitive and reflexive over S_Q
- How to handle information flow?
  - Define a partially ordered set containing quasi-ordered set
  - Add least upper bound, greatest lower bound to partially ordered set
  - It's a lattice, so apply lattice rules!

In Detail …

- $\forall x \in S_Q$: let $f(x) = \{ y \mid y \in S_Q \wedge y \leq_Q x \}$
  - Define $S_{QP} = \{ f(x) \mid x \in S_Q \}$
  - Define $\leq_{QP} = \{ (x, y) \mid x, y \in S_Q \wedge x \subseteq y \}$
    - S_{QP} partially ordered set under $\leq_{QP}$
    - f preserves order, so $y \leq_Q x$ iff $f(x) \leq_{QP} f(y)$
- Add upper, lower bounds
  - $S_{QP}' = S_{QP} \cup \{ S_Q, \varnothing \}$
  - Upper bound $ub(x, y) = \{ z \mid z \in S_{QP} \wedge x \subseteq z \wedge y \subseteq z \}$
  - Least upper bound $lub(x, y) = \cap ub(x, y)$
    - Lower bound, greatest lower bound defined analogously

And the Policy Is …

- Now $(S_{QP}', \leq_{QP})$ is lattice
- Information flow policy on quasi-ordered set emulates that of this lattice!