Information Flow Control (IFC)
Introduction

- Information flow control
  - form of *mandatory* control for security
  - important security properties like information confidentiality and integrity are information flow properties
  - security by design
  - compositional
  - end-to-end security guarantees

Information Flow Control (IFC)
Basics

- Information flow control works as follows
  - assign labels to subjects and objects for tracking the flow of information in the system
  - define access rules (read and write) in terms of can-flow-to relation on labels
- Labels play a crucial role in IFC systems
- One of the main challenges for IFC systems is user acceptance
  - hindered by current complicated label models

IFC

- Bell La Padula 1974,
- Denning – 1975
- Biba - 1976

Lattice Model

- Lattice: consists of a finite partially ordered set together with a least upper bound and greatest lower bound operator on the set.
- Policy: information is permitted to flow from a lower class to upper class.

Lattice Model

- Lattice FM = < S, O, SC, F, ⊕, ⊗, → >
- S: set of subjects
- O: set of objects
- SC: finite set of security classes
- F: mapping function from S or O to SC, object O is bound to a class called security classification, subject S is bound to a class called security clearance

⊕: Least upper bound operator on SC

⊗: Greatest lower bound operator on SC

→Flow relation on pairs of security classes

FM is considered as secure only if the execution of a sequence of operations cannot cause an information flow that violates the relation →

→ reflexive, transitive, anti-symmetric for all A,B,C Ɛ SC.

Reflexive: A → A

- **Information flow from an object to another object at the same class does not violate security.**

Transitive: A → B and B → C ➔ A → C .

- **Valid flow does not necessarily occur between two adjacent classes**

Anti-symmetric: A → B & B → A ➔ A=B

- **If information can flow back and forth between two objects, they must have the same class**

***********Properties ******

Aggregation: A → C and B → C implies A U B → C

- **If information can flow from both A & B to C, information aggregate of A & B can flow to C.**

Separation: A U B → C implies A → C and B → C

- **If the information aggregate of A & B can flow to C, information can flow from either A or B to C**

Lattice (contd)

- Example: Linear ordered lattice
- SC = {C1, ..., Cn}, Ci→Cj iff i <= j
- Ci $\oplus$ Cj = C{max(i,j)}
- Ci $\otimes$ Cj = C{min(i,j)}
- C1 → C2 → C3 → ... → Cn-1 → Cn
- Information can only flow upward, and once it reaches to a class Ci, it cannot flow down to any class below Ci
- Suitable for any system in which all classes need to be totally ordered

Information Flows

- Channels - mechanisms for signalling information
- Explicit Flows:
  - X:=Y – Y flows to X
- Covert Channels - primary purpose is not information transfer
- Implicit Flow:

  h:= h mod 2;

  l:=0;

  if h=1 then l:=1

  else skip

- Does not leak the exact value of i to l , but it does leak some information about the value of h to l
- Someone observing lo could tell whether hi is negative or not.

high ← low

(secret) ↛ (public)

EXPLICIT FLOW

```
function test (bool high)
    bool low;
    low = high;
```

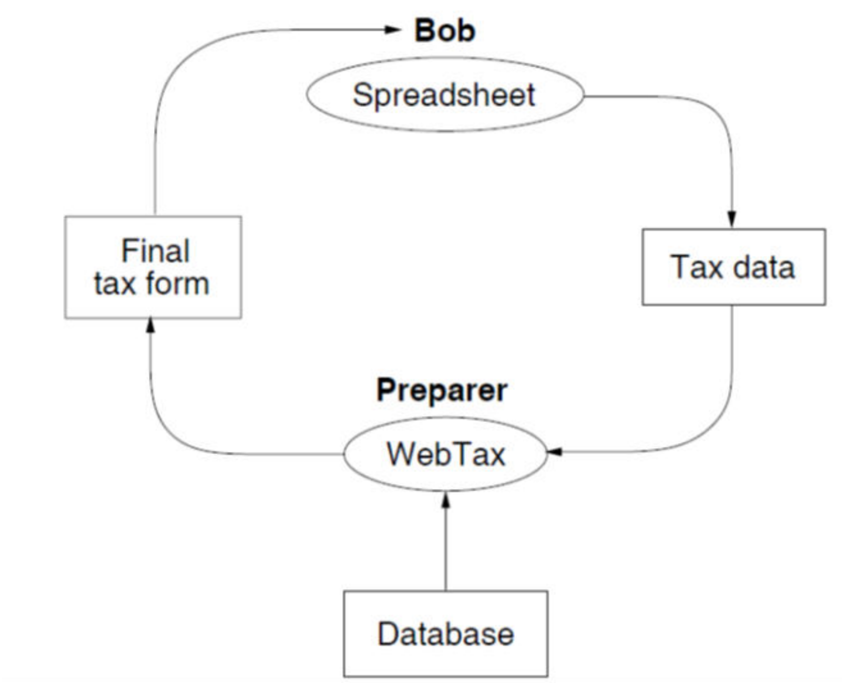
IMPLICIT FLOW

```
function test (bool  high)
    bool low = 0;
    if high = 1
        low = 1;
```

COVERT FLOW

```
function test (bool  high)
    bool low = 0;
    while high = 0;
    low = 1;
```

Confidentiality Example



- Principal Preparer – distributor of WebTax-may have privacy interests
- WebTax application computes the final tax form using a proprietary database, shown at the bottom (owned by Preparer).
  - this might, contain secret algorithms for minimizing tax payments.
  - Since this principal is the source of the WebTax software, it trusts the program not to distribute the proprietary database through malicious action,
  - However, the program might leak information because it contains bugs.

Vickery Auction

```
nt{⊥ → ⊥; A ← au ⊓ B ← au} winner[10];
nt{⊥ → ⊥; A ← au ⊓ B ← au} i;
or (i = 1..10) {
    int{A → au; A ← au ⊓ B ← au} bidA =
    getAliceBid(i);
    int{B → au; A ← au ⊓ B ← au} bidB = getBobBid(i);
    // end of auction i

    int{⊥ → ⊥; A ← au ⊓ B ← au} openA =
    declassify(bidA, {⊥ → ⊥; A ← au ⊓ B ← au});
    int{⊥ → ⊥; A ← au ⊓ B ← au} openB =
    declassify(bidB, {⊥ → ⊥; A ← au ⊓ B ← au});

    // compute winner

    winner[i] = computeWinner(openA, openB);
    // process payment of winning bid
```

Vickrey Auction Example

Autioneer is a trusted party with label:
$AU^{(AU, \{A,B,AU\}, \{AU\})}$

$bidA^{(A, \{A,AU\}, \{A\})}$, $bidB^{(B, \{B, AU\}, \{B\})}$

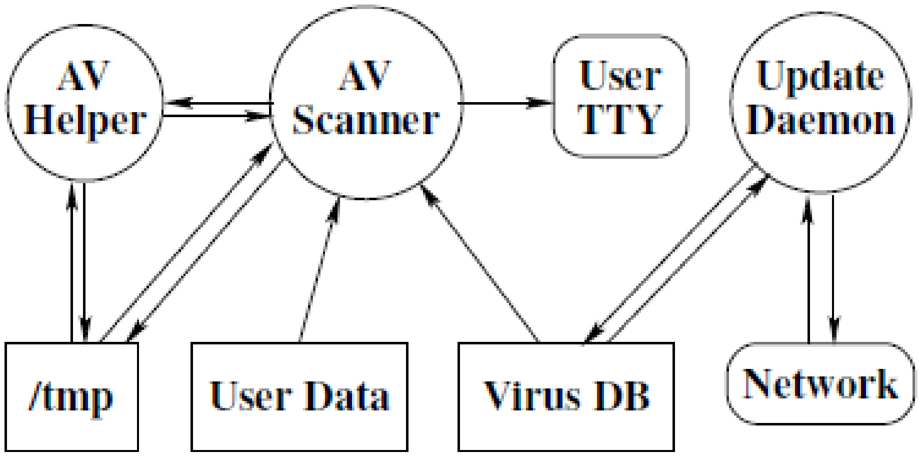
$AU \leftarrow bidA$; $AU^{(AU, \{A,AU\}, \{A, AU\})}$ //reads bi
$AU \leftarrow bidB$; $AU^{(AU, \{AU\}, \{A, B, AU\})}$ //reads bi

Now $AU^{(AU, \{AU\}, \{A, B, AU\})}$ is declassified to
$winner^{(AU, \{A,B,AU\}, \{A, B, AU\})}$

Reading rule:
If $s^{(S1, R1, W1)}$ reads a value $o^{(S2,R2,W2)}$ the
new label of 's' is $s^{(S1, R1 \cap R2, W1 \cup W2)}$

TAKEAWAY: IN CASE OF RWFM THE INFORMATION CANNOT BE DISCLOSED TO ENTITIES WHO HAVE NOT INFLUENCED THE DATA.

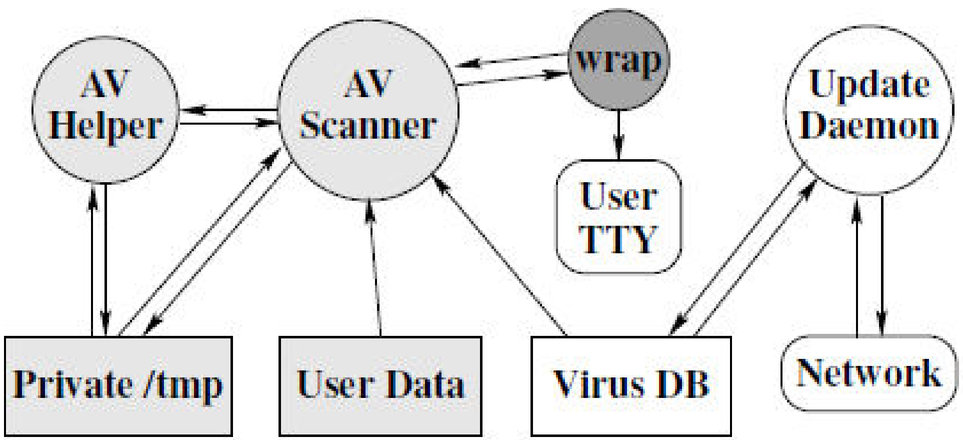
Enforcing data security policy while executing untrusted code



- Lightly Shaded – Confidential
- Unshaded – non-confidential
- Dark Shaded – Special privileges

to relay the scanner's confidential output to the terminal.

- Circles: Processes
- Rectangles: Files/Dir
- Rounded Rect: Devices



Conference Systems

- Lambda–Chair
- EasyChair
- HotCRP

State of the Art

- Centralized labels – Denning (1975)
- Decentralized Model – Myers and Liskov (1997)
- Robust Declassification (2004)
- Flume (2007), Laminar(2012), Histar OS(2006)

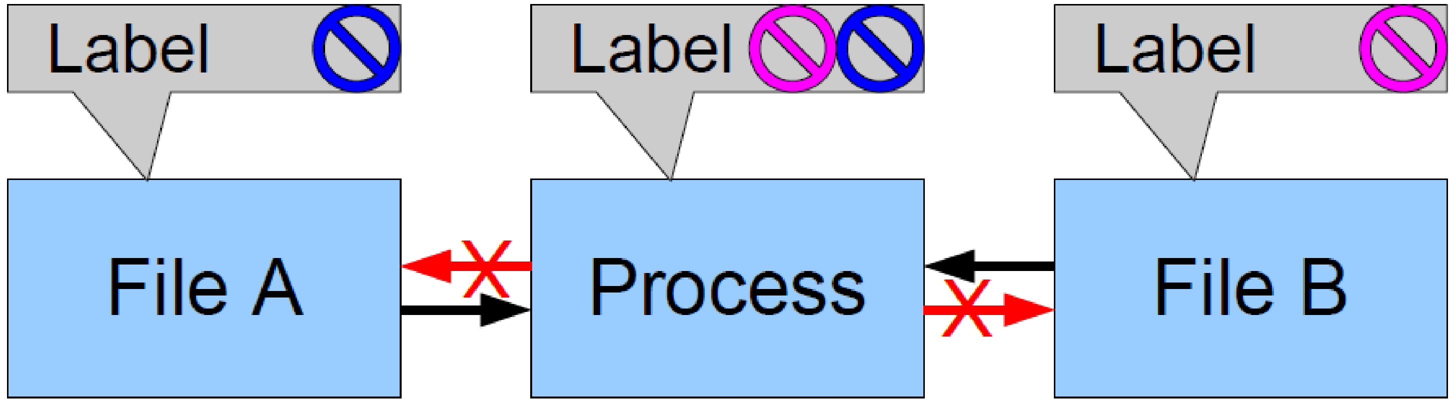
Decentralized Label Model

Myers and Liskov (2000)

- addresses the weaknesses of earlier approaches to the protection of confidentiality in a system containing untrusted code or users, even in situations of mutual distrust
- allows users to control the flow of their information without imposing the rigid constraints of a traditional MLS
- defines a set of rules that programs must follow in order to avoid leaks of private information
- protects confidentiality for users and groups rather than for a monolithic organization
- introduces a richer notion of declassification
  - in the earlier models it was done by a trusted subject; in this model principals can declassify their own data

Labels control information flow

- Color is category of data (e.g. my files)
- Blue data can flow only to other blue objects



Drawbacks of State-of-the-art

- 1985 Trusted Computer Systems Evaluation Criteria (Orange Book)
  - defines the security of a computer system by how well it implements flow control and how good its assurance is
- Despite huge efforts, systems developed had several drawbacks:
  - large TCB, slow, not easy to use, and very limited functionality

Drawbacks of State-of-the-art

- 2000 Myers & Liskov (DLM) and robust declassification (2004
  - only readers for protecting confidentiality and only writers for protecting integrity
  - Essentially becomes DAC due to free Declassification
  - Flaw: *for a proper tracking of any information flow property, it is important to control both reading and writing by subjects*

Drawbacks of State-of-the-art

- HiStar, Flume and Laminar systems
  - based on the product of Confidentiality and Integrity
  - Flaw: *confidentiality and integrity are not orthogonal properties*
  - *The declassification rules essentially becomes discretionary*
  - Fred Schneider, in his book[#] chapter, clearly brings out the perils of combining confidentiality and integrity policies in this manner

yet to be published,
available at http://www.cs.cornell.edu/fbs/publications/chptr.MAC.pdf

Drawbacks of State-of-the-art

- 2012 Mitchell et al. (DC labels)
  - not easy to derive consistent DC labels for modelling a given requirement
  - Flaw: *support for downgrading (discretionary control) is orthogonal to the IFC, thus, defeating the purpose of the mandatory controls*

Drawbacks of State-of-the-art

- 2011 Butler Lampson in HiStar technical perspective
  - *This is the latest step in the long and frustrating journey toward secure computing. It is a convincing solution for some serious practical problems. The general-purpose computing that failed in the 1980s has not been tried*

RWFM Model

Narendra kumar, RKS 2014

Readers-Writers Labels

- Security requirements of practical applications are often stated / easily understood in terms of who can read / write information
- Observations:
  - information readable by s_1 and $s_{2,}$ can-flow-to information readable only by s_1
  - information writable only by $s_{1,}$ can-flow-to information writable by s_1 and s_2
- Readers and writers can be used as labels!!

RWFM Label Format

- (owner/authority, readers, writers)
  - First component is a single subject denoting
    - *owner* in case of an object label
    - *authority* in case of a subject label
  - Second component is a set of subjects denoting
    - permissible readers in case of an object label
    - subjects who can read all the objects that this subject can read in case of a subject label
  - Third component is a set of subjects denoting
    - permissible writers in case of an object label
    - subjects who can write all the objects that this subject can write in case of a subject label

Permissible Flows in RWFM

- Given any two RW classes $RW_1=(s_1,R_1,W_1)$ and $RW_2=(s_2,R_2,W_2)$, information is allowed to flow from RW_1 to RW_2, denoted $RW_1 \leq RW_2$ only if $R_1 \supseteq R_2$ and $W_1 \subseteq W_2$. Formally

$$\frac{R_1 \supseteq R_2 \qquad W_1 \subseteq W_2}{(s_1,R_1,W_1) \leq (s_2,R_2,W_2)}$$

Join and Meet of RW Classes

- Let $RW_1=(s_1,R_1,W_1)$ and $RW_2=(s_2,R_2,W_2)$, be any two RW classes. Their join ($\oplus$) and meet ($\otimes$) are defined as follows:

$$(s_1,R_1,W_1) \oplus (s_2,R_2,W_2) = (s_3,R_1 \cap R_2,W_1 \cup W_2)$$

$$(s_1,R_1,W_1) \otimes (s_2,R_2,W_2) = (s_3,R_1 \cup R_2,W_1 \cap W_2)$$

RW Classes form a Bounded Pre-Lattice

- **Prop**: The relation $\leq$ on RW classes is reflexive and transitive i.e., **a pre-order**
- **Theorem**: The set of all RW classes $SC_{RW}=S\times 2^S\times 2^S$, together with the ordering $\leq$, join $\oplus$ and meet $\otimes$ form a **bounded pre-lattice**. For $s\in S$, $(s,S,\emptyset)$ denotes a minimum element and $(s,\emptyset,S)$ denotes a maximum element.

Readers-Writers Flow Model

- Above theorem establishes the soundness of RW classes w.r.t. Denning's model i.e., suitability of RW classes for studying information flow properties in a system
- **Readers-Writers Flow Model (RWFM)** is defined as a six-tuple $(S, O, SC_{RW}, \leq_{RW}, \oplus_{RW}, \otimes_{RW})$, where S is the set of subjects and O is the set of objects in an information system, and SC_{RW}, $\leq_{RW}, \oplus_{RW}, \otimes_{RW}$ are as defined previously

Notation

- Flow model together with a labelling function defines an access policy
- Labelling function $\lambda : S \cup O \rightarrow SC_{RW}$
- $A_\lambda(e)$, $R_\lambda(e)$ and $W_\lambda(e)$ denote the first, second and third components of $\lambda(e)$
- λ is omitted when clear from the context
- For a subject s, $A(s)=s$

Access Rules in RWFM

- Given a RWFM and functions A, R and W describing a labelling,
  - A subject s is allowed to read an object o if
    - $A(s) \in R(o)$ **and** $R(o) \supseteq R(s)$ **and** $W(o) \subseteq W(s)$
  - A subject s is allowed to write an object o if
    - $A(s) \in W(o)$ **and** $R(s) \supseteq R(o)$ **and** $W(s) \subseteq W(o)$

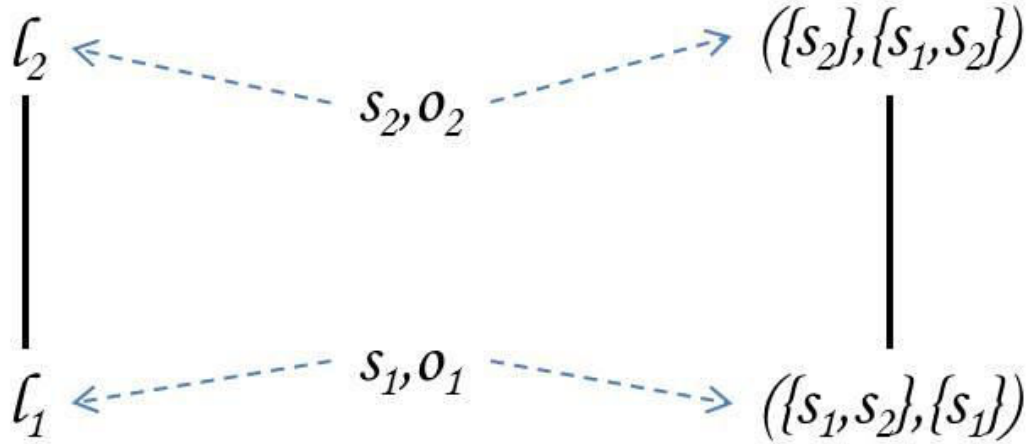
DAC MAC DAC + MAC

Completeness of RWFM w.r.to Denning

- **Theorem**: Given a Denning's flow model DFM = $(S,O,SC,\leq,\oplus)$ and a policy $\lambda : S \cup O \rightarrow SC$, there exists a labelling $\lambda_{RW} : S \cup O \rightarrow SC_{RW}$, in the RWFM that enforces the same policy i.e.,

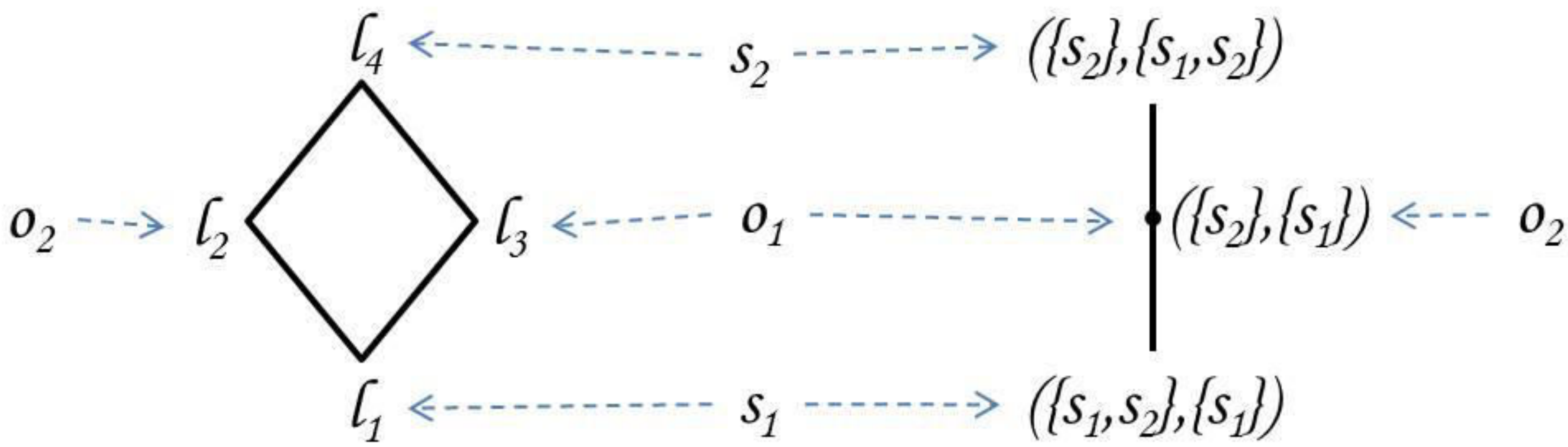
1. s is permitted to read o by Denning's policy if and only if it is permitted by RW-policy
2. s is permitted to write o by Denning's policy if and only if it is permitted by RW-policy

Illustrative Examples



Denning's Policy   Readers-Writers Policy

Illustrative Examples (contd)



Denning's Policy

Readers-Writers Policy

State of an Information System

- State of an information system is defined as the set of subjects and objects in the system together with their labels. Initial state
  - Objects and their labels as required for application
  - Each subject s starts with label $(s,*,\phi)$
- Whenever a subject tries to perform an operation on an object, it may lead to a state change and will have to be permitted only if deemed safe
  - Read
  - Write
  - Create
  - Downgrade
  - Relabel

State Transitions in RWFM

- Subject s with label (s_1,R_1,W_1) requests *read* access to an object o with label (s_2,R_2,W_2)
  - If $s_1 \in R_2$ then
    - relabel s to $(s_1, R_1 \cap R_2, W_1 \cup W_2)$ and ALLOW access
  - Else
    - DENY access
- POSSIBLE state change (label of s may change)

s can

s has accessed information accessible only by

s is influenced by both W_1 and W_2

State Transitions in RWFM

- Subject s_1 with label (s_1, R_1, W_1) requests <u>*write*</u> ... $(s_2,$ ...
  - If $s_1 \in W_2$ and $R_1 \supseteq R_2$ and $W_1 \subseteq W_2$ then
    - ALLOW access
  - Else
    - DENY access
- <u>NO</u> state change

s can write o

all subjects ... can access ...

all subjects that have influenced the current information of s can also influence o

State Transitions in RWFM

s, and all subjects that have influenced the current information of s have influenced o

accessed by s so far, ca

- Subject s with label (s,R,W) requests *creation* of an object o
  - create an object o and label it (s,R,W$\cup${s})
- DEFINITE state change (a new object is added to the system)

State Transitions in RWFM

- Subject s w… an …

subjects that can access o, can also access

subjects that could not access o but can access its downgraded version must have influenced information in o

all th… access its downgra… on also

… $W_3)$

  - If s_1 … and $s_1 = s_2$ … and $W_1 = W_2 = W_3$ and $R_1 = R_2$ and $R_3 \supseteq R_2$ and $R_3 - R_2 \subseteq W_2$ then
    - ALLOW
  - Else
    - DENY
- <u>POSSIBLE</u> state change (label of o may change)

State Transitions in RWFM

- Subject s w

s, and all subjects that influenced the current information of s have influenced the relabelling

all subjects that can access the relabelled object, could have accessed all the information that s has accessed so far, and the original object

rela $s_3, R_3,$

  - If $s_1 \in R_2$ and $s_1=s_2=s_3$ and $W_2 \subseteq W_1$ and $W_3=W_1 \cup \{s\}$ and $R_2 \supseteq R_1 \supseteq R_3$ then
    - ALLOW
  - Else
    - DENY
- POSSIBLE state change (label of o may change)

Downgrading (Declassifying)

- For practical applications, adding readers (downgrading) to the result of a computation is essential for use by relevant parties
- Downgrading rules
  - only the owner of information may downgrade it
  - if a single source is responsible for the information, then readers that can be added is unrestricted
  - if multiple sources influenced the information, then only those who influenced it may be added as readers

Reasoning about Information Flow between Objects in RWFM (1)

- **Theorem 1**: Information in object o_1 with label (s_1,R_1,W_1) cannot flow to object o_2 with label (s_2,R_2,W_2) if any of the following conditions hold:

1. $R_1=\emptyset$
2. $W_2=\emptyset$
3. $W_1 \not\subseteq W_2$
4. $R_2 \not\subseteq (R_1 \cup W_1 \cup W_2)$

Reasoning about Information Flow between Objects in RWFM (2)

- **Theorem 2**: If $R_2 \subseteq R_1$ and $R_1 \cap W_2 \neq \emptyset$, and none of the conditions in Theorem 1 hold, only a subject in R_1 can make information to flow from o_1 to o_2.
- **Theorem 3**: If $R_2 \subseteq (R_1 \cup W_1)$ and $(R_1 \cup W_1) \cap W_2 \neq \emptyset$, and none of the conditions in Theorems 1 and 2 hold, information can flow from o_1 to o_2 only as a result of a collusion between a subject in R_1 with a subject in W_1.

(help us identify the only possible culprits in the case of an info. flow.)

Reasoning about Information Flow between Objects in RWFM (3)

- **Theorem 4**: If none of the conditions in Theorems 1, 2 and 3 hold, information can flow from o_1 to o_2 only as a result of a collusion between a subject in R_1 with all the subjects in $R_2 \cap W_2$.

Information Flow between entities in RWFM

- **Theorem**: Given a Denning's flow model DFM = $(S,O,SC,\leq,\oplus)$ with a policy $\lambda : S \cup O \rightarrow SC$, and the corresponding policy in the RWFM (constructed in the completeness theorem), the following holds: "information can flow from entity e_1 to entity e_2 under Denning's policy if and only if it can flow without downgrading in the RW-policy", where entity is either a subject or an object in the system.

Informally

- While the completeness theorem proved that "immediate info flows" (flows resulting due to a single operation by subjects) in a Denning's policy can be simulated by the corresponding RW policy, this theorem says that all info flows (in single or multiple steps between not only a subject and an object, but between any two entities) in a Denning's policy can be simulated in the RW policy modulo downgrading.

Relations among subjects in RWFM

- **Prop**: Let DFM with λ be a Denning's policy, and let A, R and W denote the corresponding labelling in the RWFM (constructed in the completeness theorem). For any two subjects s_1 and s_2, the following holds:

1. $s_1 \in R(s_2)$ if and only if $R(s_2) \supseteq R(s_1)$
2. $s_1 \in W(s_2)$ if and only if $W(s_1) \subseteq W(s_2)$

Subject dominance relations in RWFM

- Subject s_1 "***read dominates***" s_2, $s_2 \leq_R s_1$, if $s_1 \in R(s_2)$
- Subject s_1 "***write dominates***" s_2, $s_2 \leq_W s_1$, if $s_1 \in W(s_2)$
- Subject s_1 "***information dominates***" s_2, $s_2 \leq_I s_1$, if $s_2 \leq_R s_1$ and $s_1 \leq_W s_2$
- **Theorem**: All the dominance relations on subjects are reflexive and transitive (pre-order)

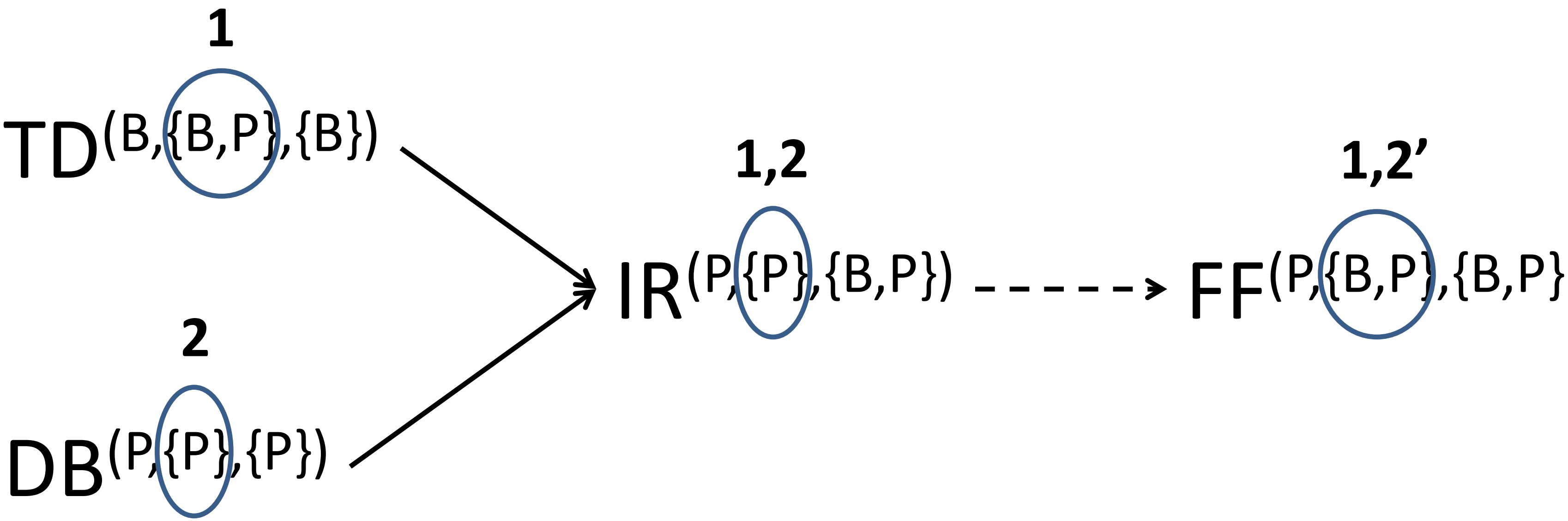
Principal hierarchy vs subject dominance

- The standard notion of principal hierarchy can be captured as follows
  - Given subjects s_1 and s_2, we say that s_1 dominates s_2 in the principal hierarchy written $s_2 \leq s_1$, if $s_2 \leq_R s_1$ and $s_2 \leq_W s_1$
- Considering the fact that information flows in opposite directions in reading and writing, we recommend that **in the context of IFC, information dominance provides a better notion of subject superiority than principal hierarchy**

Example-1
WebTax

- Bob provides his tax-data to a professional tax preparer, who computes Bob's final tax form using a private database of rules for minimizing the tax payable and returns the final form to Bob
- Security requirements
  1. Bob requires that his tax-data remains confidential
  2. Preparer requires that his private database remains confidential

Example-1
WebTax



TD	Tax-data	IR	Intermediate results
DB	Database of tax optimization rules	FF	Final tax form
⟶	Flows-to	- ->	Downgraded-to

Example-1
WebTax

	DLM	DC	RWFM
TD	{B: B}	(B, B)	(B, {B,P}, {B})
DB	{P: P}	(P, P)	(P, {P}, {P})
IR	{B: B; P: P}	($B \wedge P$, $B \vee P$)	(P, {P}, {B,P})
FF	{B: B}	(B, $B \vee P$)	(P, {B,P}, {B,P})

- <u>DLM label format</u>: policies separated by ';', where each policy is of the form 'owner: readers'
- <u>DC label format</u>: 'readers, writers', where readers control confidentiality, writers control integrity
- <u>RWFM label format</u>: 'owner, readers, writers'

DLM, DC and RWFM Comparison

	DLM	DC	RWFM
Confidentiality	only Readers	only Readers	Readers and Writers
Integrity	only Writers	only Writers	Readers and Writers
Downgrading (DAC)	Purely discretionary	Purely discretionary	Consistent with IFC (MAC)
Ownership	Explicit	Implicit	Explicit
Authority	Orthogonal to the label	Orthogonal to the label	Explicit in the label

DLM, DC and RWFM Comparison

	DLM	DC	RWFM
Principal hierarchy and Delegation	Orthogonal to the label	Orthogonal to the label	Embedded in the label
Bi-directional flow	Difficult	Difficult	Simple and Accurate
Ease of use	Moderate	Moderate	Easy
Label size	Moderate to Large	Large	Small
No. of labels	Large	Large	Small (as required by the application)

Readers-Writers Label Model

Advantages

- Labels are intuitive / easy to understand
- Automatic extraction of labels from security requirements
- Efficient label manipulations
- Easy to verify / validate required security properties

Illustrative Examples

Information misuse detection using RWFM

Example - 1

```
l := T
t := F
if h then
    t := T
if ¬t then
    l := F
```

- Benchmark program for evaluating soundness of flow-sensitive dynamic labelling analysis
- The challenge is to track the indirect information flow from h to l
  - *l must be labelled sensitive when h is sensitive*

Analysis – two-point lattice

Execution Context:

$$P = S = \{\text{Lo}, \text{Hi}\}; V = \{h, l, t\}; G = \{h\}; p = \text{Hi}$$

		λ_{LH}			
		h	pc	l	t
-1		H	?	?	?
0	$l :=$ T	H	L	L	L
1	$t :=$ F	H	L	L	L
2	`if` h `then`	H	L	L	L
3	$\quad t :=$ T	H	H	L	H
4	`if` $\neg t$ `then`	H	H	L	H
5	$\quad l :=$ F	H	H	H	H
6		H	H	H	H

Analysis – RWFM

Execution Context:

$$P = S = \{\text{Lo}, \text{Hi}\}; V = \{h, l, t\}; G = \{h\}; p = \text{Hi}$$

		$\lambda_{\mathtt{RW}}$			
		h	pc	l	t
-1		(Hi,{Hi},{Lo,Hi})	?	?	?
0	$l := \mathtt{T}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})
1	$t := \mathtt{F}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})
2	$\mathtt{if}\ h\ \mathtt{then}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Lo,Hi},{Hi})
3	$\quad t := \mathtt{T}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Hi},{Lo,Hi})
4	$\mathtt{if}\ \neg t\ \mathtt{then}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Lo,Hi},{Hi})	(Hi,{Hi},{Lo,Hi})
5	$\quad l := \mathtt{F}$	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})
6		(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})	(Hi,{Hi},{Lo,Hi})

Summary

- Since Lo$\notin$R(λ_6(l)), Lo will not be allowed to observe the value of l at point 6
  - our analysis correctly marked the flow of information from h to l, irrespective of whether the assignments at points 3, 5 were executed or not
- Similarly, Lo will not be allowed to observe the value of t or the status of the program - like termination, execution time, resource usage etc. - beyond point 2
- RWFM analysis more finer-grained than analysis using the two-point syntactic lattice
  - distinctions become more clear in non-trivial lattices

Example - 2

- **Password update program**
  - v_1, v_2 and v_3 denote password, guess and new password respectively,
  - C denotes the client whose password is to be updated, and
  - L denotes the system admin responsible for updating the password

Analysis – RWFM

Execution Context:

$P = S = \{L, C\}; V = \{v_1, v_2, v_3, v_4\}; G = \{v_1, v_2, v_3\}; p = L$

$\lambda(v_1) = (L, \{L\}, \{L, C\})$

$\lambda(v_2) = \lambda(v_3) = (C, \{L, C\}, \{C\})$

		pc	v_4
-1		?	?
0	`if` $(v_1 == v_2)$ `then`	$(L, \{L, C\}, \{L\})$	$(L, \{L, C\}, \{L\})$
1	$v_1 := v_3$	$(L, \{L\}, \{L, C\})$	$(L, \{L\}, \{L, C\})$
2	$v_4 :=$ T	$(L, \{L\}, \{L, C\})$	$(L, \{L\}, \{L, C\})$
3	`else` $v_4 :=$ F	$(L, \{L\}, \{L, C\})$	$(L, \{L\}, \{L, C\})$
4	`return` v_4 `to` C	$(L, \{L\}, \{L, C\})$	$(L, \{L\}, \{L, C\})$
5		$(L, \{L\}, \{L, C\})$	$(L, \{L, \boxed{C}\}, \{L, C\})$

Downgraded

Analysis – DIFC

$\texttt{endorse}(v_2, v_3)$

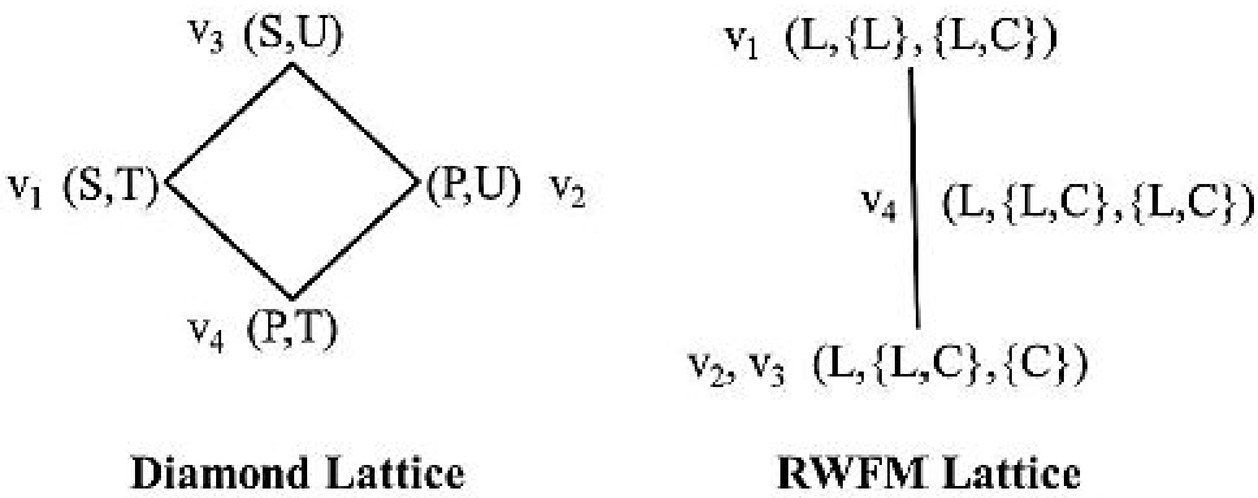
$\quad \texttt{if}\ (\texttt{declassify}(v_1 == v_2))\ \texttt{then}$

$\qquad v_1 := v_3$

$\qquad v_4 := \mathrm{T}$

$\quad \texttt{else}\ v_4 := \mathrm{F}$

Comparison – RWFM vs DIFC



- Note that the flows v_2 to v_1, v_2 to v_4 and v_3 to v_1 seem natural and easier to visualize in the RWFM lattice
- Impossible in the diamond lattice without declassify and endorse !! True not only in this example, but in the general case as well

Drawbacks of the diamond lattice approach

- Under the reasonable assumption that attackers are assigned label (P,U), and trusted subjects are assigned label (S,T), no non-trivial secure computation is possible without endorsing attackers inputs and declassifying the secure outputs
  - having declassify and endorse as explicit language constructs opens up a lot of covert channels which are impossible to overcome

Example - 3

- **Scheduling a meeting time**
  - Mutually distrusting parties Alice (denoted by p_1) and Bob (denoted by p_2) wish to schedule a joint meeting using a third party scheduler denoted p_3
  - Alice and Bob's calendars are denoted by c_a and c_b labelled $(p_1, \{p_1, p_3\}, \{p_1\})$ and $(p_2, \{p_2, p_3\}, \{p_2\})$ respectively

Analysis – RWFM labels

Execution Context:

$$P = S = \{p_1, p_2, p_3\}; V = \{c_a, c_b, m\}; G = \{c_a, c_b\}; p = p_3$$

		λ_{RW}	
		pc	m
-1		?	?
0	$m := c_a$ op c_b	$(p_3, \{p_1, p_2, p_3\}, \{p_3\})$	$(p_3, \{p_1, p_2, p_3\}, \{p_3\})$
1	return m to p_1	$(p_3, \{p_3\}, \{p_1, p_2, p_3\})$	$(p_3, \{p_3\}, \{p_1, p_2, p_3\})$
2	return m to p_2	$(p_3, \{p_3\}, \{p_1, p_2, p_3\})$	$(p_3, \{\boxed{p_1}, p_3\}, \{p_1, p_2, p_3\})$
3		$(p_3, \{p_3\}, \{p_1, p_2, p_3\})$	$(p_3, \{\boxed{p_2}, p_3\}, \{p_1, p_2, p_3\})$

Meeting time downgraded to return to p_2

RWFM vs Laminar

- For achieving the same functionality and same security, Laminar uses special program constructs like security regions, explicit declassification and endorsement
  - difficult to write secure programs to achieve the desired results; to the contrary, it is easy for an attacker to abuse these features
  - further, the programmer also has the burden of explicitly annotating all the program variables

RWFM vs Laminar (1)

- For the same example (conference version), the meeting time computed by the server would have the label ⟨S(a, b), I()⟩ which means that it has secrecy tags a and b
  - inaccessible to both *Alice* and *Bob*
  - cannot be declassified by either of them
  - only way is to provide capabilities a^- and b^- to the scheduler $\Rightarrow$ he can leak the calendars of Alice and Bob by declassifying them, if he so chooses !!
- Note that the journal version does not contain these problems - except that in this case Alice would be forced to share his calendar with Bob !!

Summary

- Dynamic labelling of programs using the RWFM model provides a sound labelling scheme that enables the detection of misuse of information w.r.t confidentiality/integrity of the program
- The labelling is constructive $\Rightarrow$ enables the programmer to assure the security of specifications w.r.t the specified environment
- First work that provides a sound approach for a general lattice, and enables a blending of MAC (IFC) and DAC (controlled downgrading)

Language Based Security via Constraint Generators

procedure *copy2*(x: **integer class** $\{x\}$;	
var y: **integer class** $\{x\}$);	
"copy x to y"	
var z: **integer class** $\{x\}$;	
begin	
$z := 1$;	$Low \leq \underline{z}$
$y := -1$;	$Low \leq \underline{y}$
while $z = 1$ **do**	$\underline{z} \leq \underline{y} \otimes \underline{z}$
begin	
$y := y + 1$;	$\underline{y} \leq \underline{y}$
if $y = 0$	$\underline{y} \leq \underline{z}$
then $z := x$	$\underline{x} \leq \underline{z}$
else $z := 0$	$Low \leq \underline{z}$
end	
end	
end *copy2*	

Example:
Secure execution of the **if** statement

if $x = 1$ **then** $y := 1$

is described by

if $x = 1$
  then if $\underline{x} \le \underline{y}$ **then** $y := 1$ **else skip**
  else skip .

```
procedure copy1(x: integer; var y: integer);
  "copy x to y"
  var z: integer;
  begin
    y := 0;
    z := 0;
    if x = 0 then z := 1;
    if z = 0 then y := 1
  end
end copy1
```

If X < y is not tested, it will be insecure

Non-Interference of Type Systems

- A program p does not leak information if, for all possible start states S1 and S2 such that S1 is identical S2 when projected on low , whenever executing p in S1 terminates and results in S1’ and executing p in S2 terminates and results in S2’ , then S1’ and S2’ are congruent on low .
- Similarly for Integrity --- Equivalence needs to be defined.
- Termination –sensitive non-interference: as above with the addition “it terminates”
- Generalized forms of Non-interference for concurrent systems (Naren and RKS 2017)

Tax Example using JIF

ECURE TAX PREPARING PROGRAM

```
prepareTax authority ( Bob ) {
 boolean { Bob : Bob } optdb ;

 int {Alice:Alice; Chuck : Chuck} preparetax {Alice : Alice}( int{Alice : Alice} tax_data) w
 authority (Bob)
 {
        int tax ;
        if (( tax_data >100) && optdb )
                tax =1;
        else
                tax =0;
        return declassify ( tax , { Alice:Alice; Chuck : Chuck }) ;
 }
```