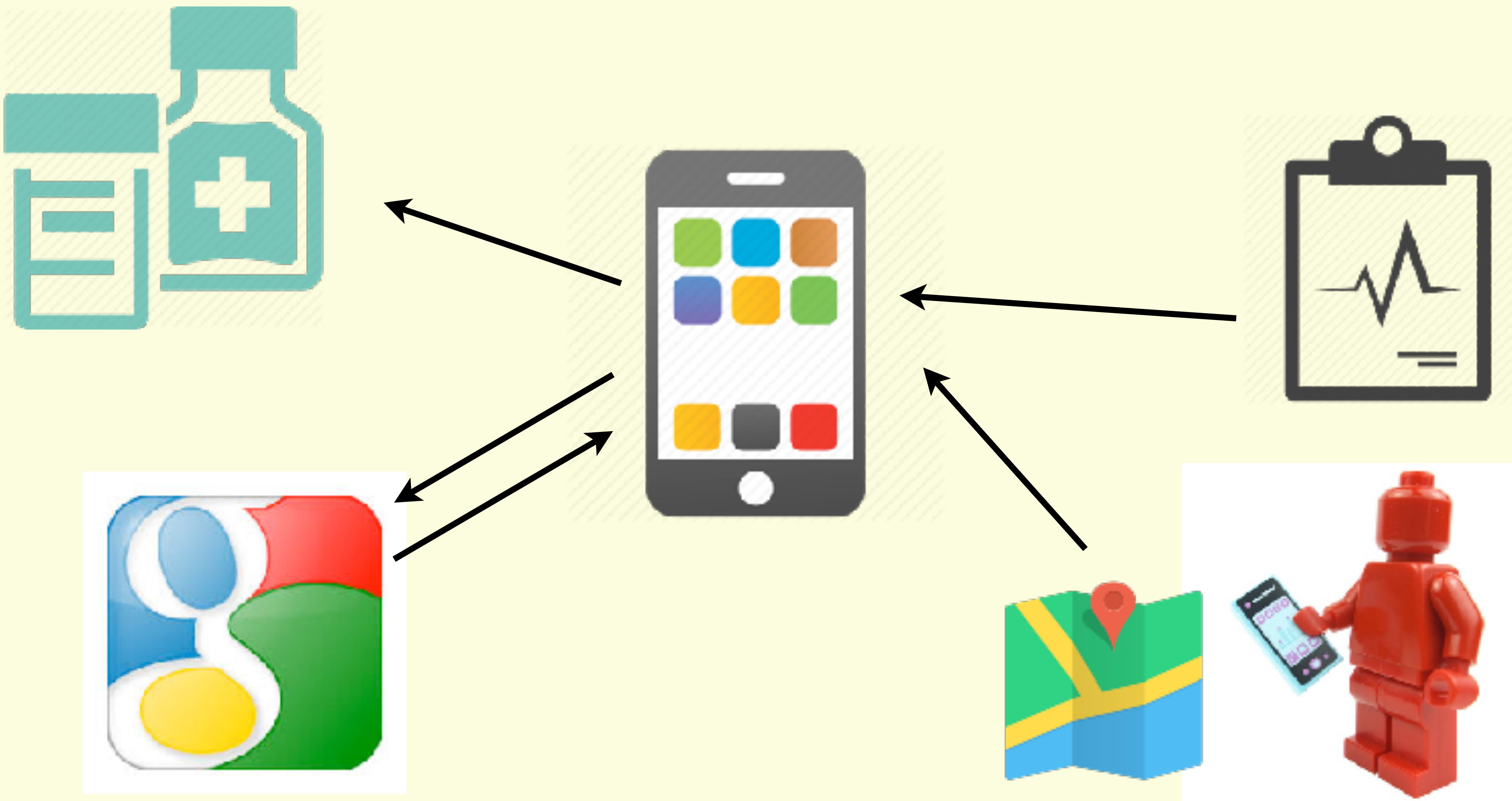
Introduction to Information Flow Security

Ana Matos
Pedro Adão Miguel Correia

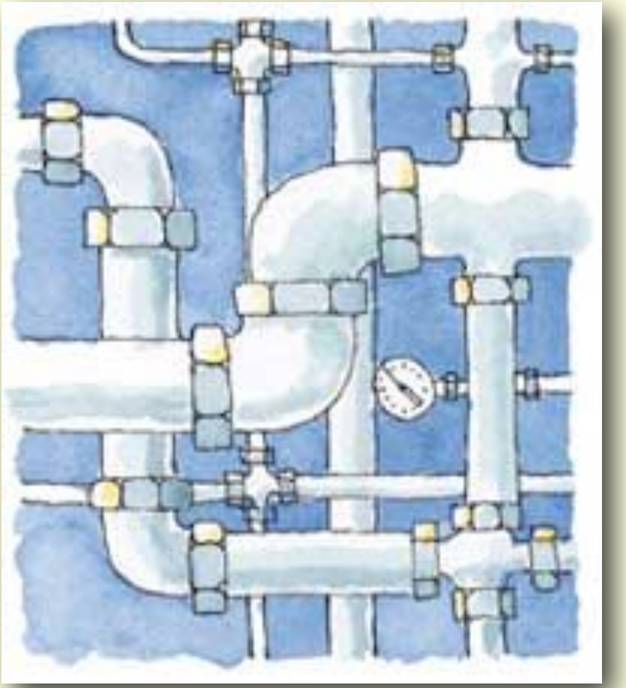
Who should access what?



“End-to-end” policies

- Confidentiality / Integrity
  Information of any level should only be readable / affected by users of levels that are the same or higher.

- Policies are respected throughout the entire process, as information flows through the system.



To think before next class

- Can we prevent illegal information flows by means os access control?
- How could we do better?

Class Outline

- Information Flow Security
  - introduction
  - Access Control to Information Flow Control
  - Encoding and exploiting information flows

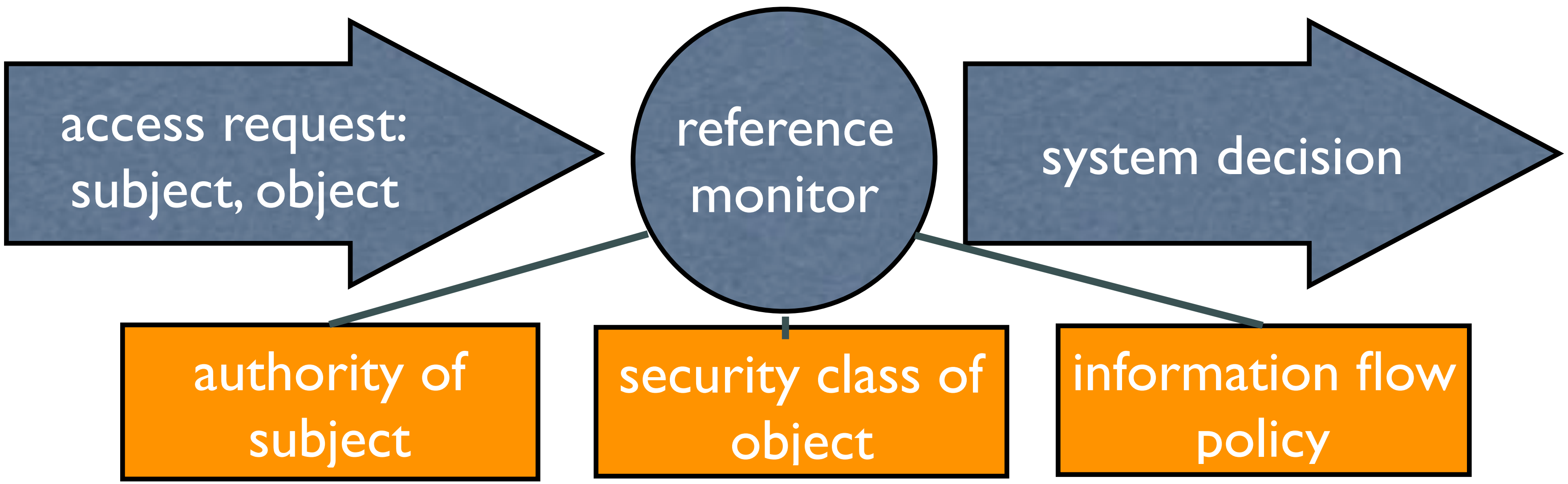
Access Control to Information Flow Control

☞ "Lattice-Based Access Control Models", R. Sandhu, 1993.

☞ "Language-Based Information-Flow Security", A. Sabelfeld and A. Myers, 2002.

Access control

- The control of interaction between subjects (active entities such as processes, ...) and objects (protected entities such as files, ...), by validating access rights of subjects to resources of the system.



- Decisions do not take into account how information is propagated by programs.

End-to-end policies

- Consider three files:
  - secret.txt (only Alice is allowed to read)
  - crucial.txt (only Bob is allowed to write)
  - public.txt, (both are allowed to read and write)
- Should Alice be allowed to copy the contents of secret.txt to public.txt? Should Bob be allowed to copy the contents of public.txt to crucial.txt?
- Can access control prevent it from happening?

Discretionary Access Control (DAC)

- Restricts access based on the identity of subjects and a set of access permissions that can be determined by subjects.
- Limitations: Access permissions might allow programs to, in effect, circumvent the policies:
  - Legally, by means of information flows that are encoded in the program.
  - Illegally, when vulnerabilities in programs and language implementations (buffer overflows, race conditions,...) can be exploited by attackers.

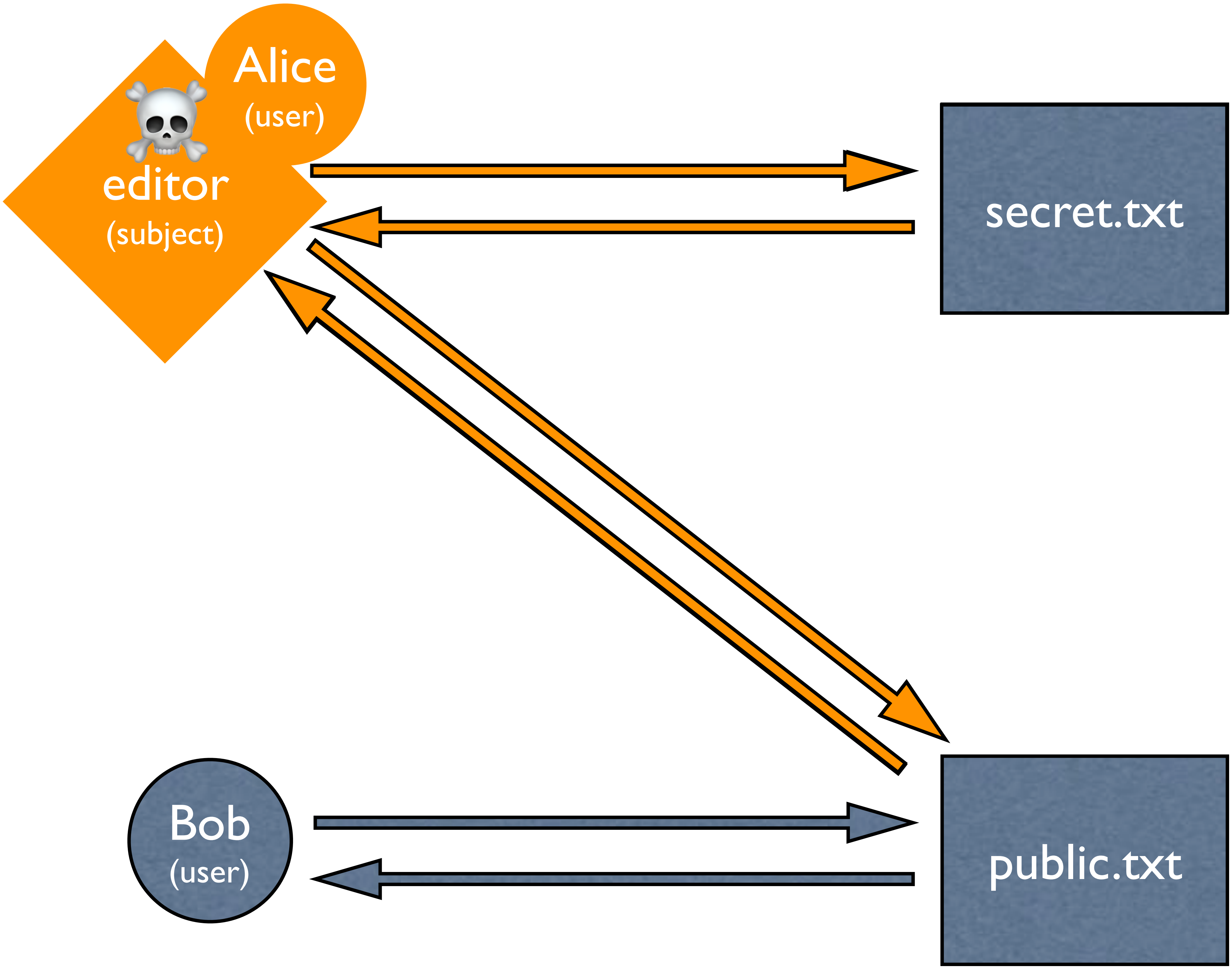
Example

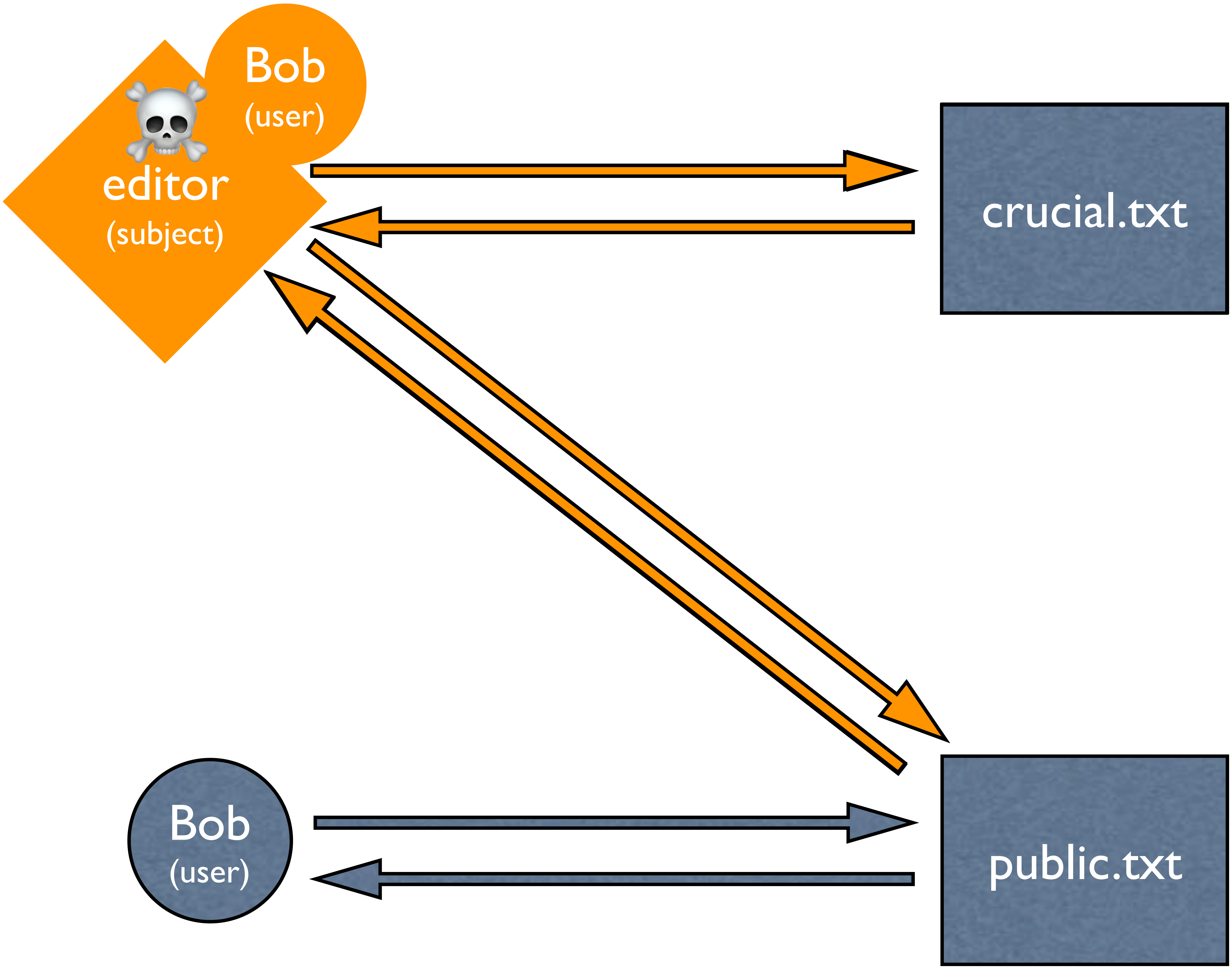
Authority of the process	secret.txt	crucial.txt	public.txt
Alice	r	-	r, w
Bob	-	w	r, w

- Can you conceive a program that discloses Alice's secret information?
  Or ruins Bob's crucial information?
  - Which owner and permissions would it have?
  - Would it require Alice's (resp. Bob's) cooperation?

Trojan Horse

- Access is validated or not depending only on the current authority and access permissions.
- Users do not always know what the program they are executing does. A program can perform unwanted things under the user's permissions.
- If a program is exploitable, the attacker can make use of its EUID (root?) privileges
- Out of reach of DAC.


Alice
(user)
editor
(subject)
secret.txt
Bob
(user)
public.txt


Bob
(user)
editor
(subject)
crucial.txt
Bob
(user)
public.txt

Mandatory Access Control (MAC)

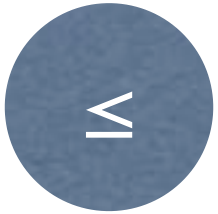
- Restricts access based on security levels of subjects (their clearances) and objects (their sensitivity). Controls how information flows in a system based on whom is performing each access.
- Example: The military hierarchy TS - S - C - U.
- Limitations: restrictiveness, covert channels.

information flow policy

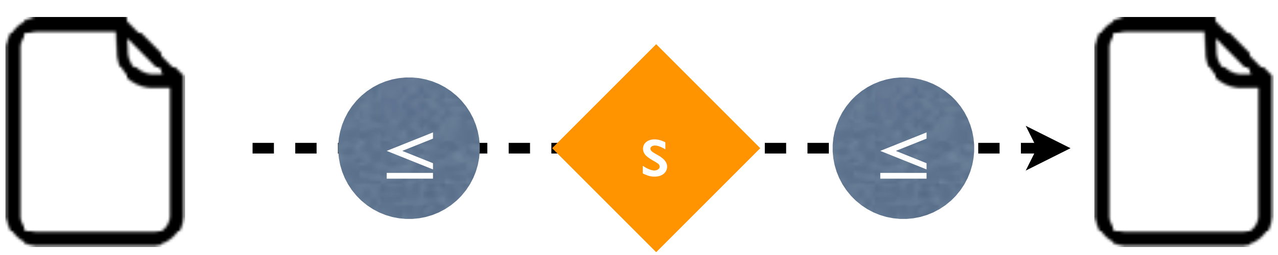
$\leq$

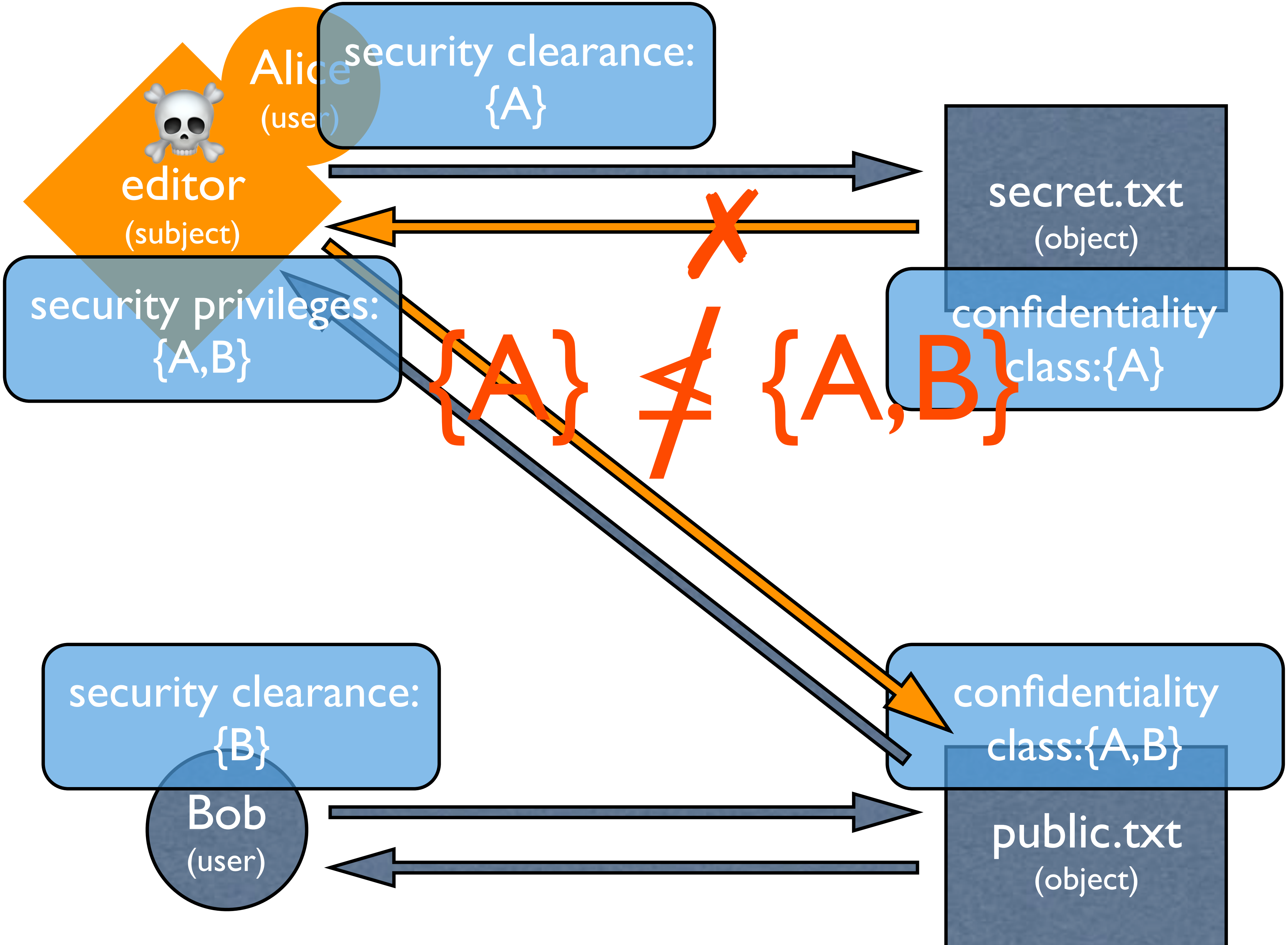
[Bell & La Padula, 1975]

Example: BLP

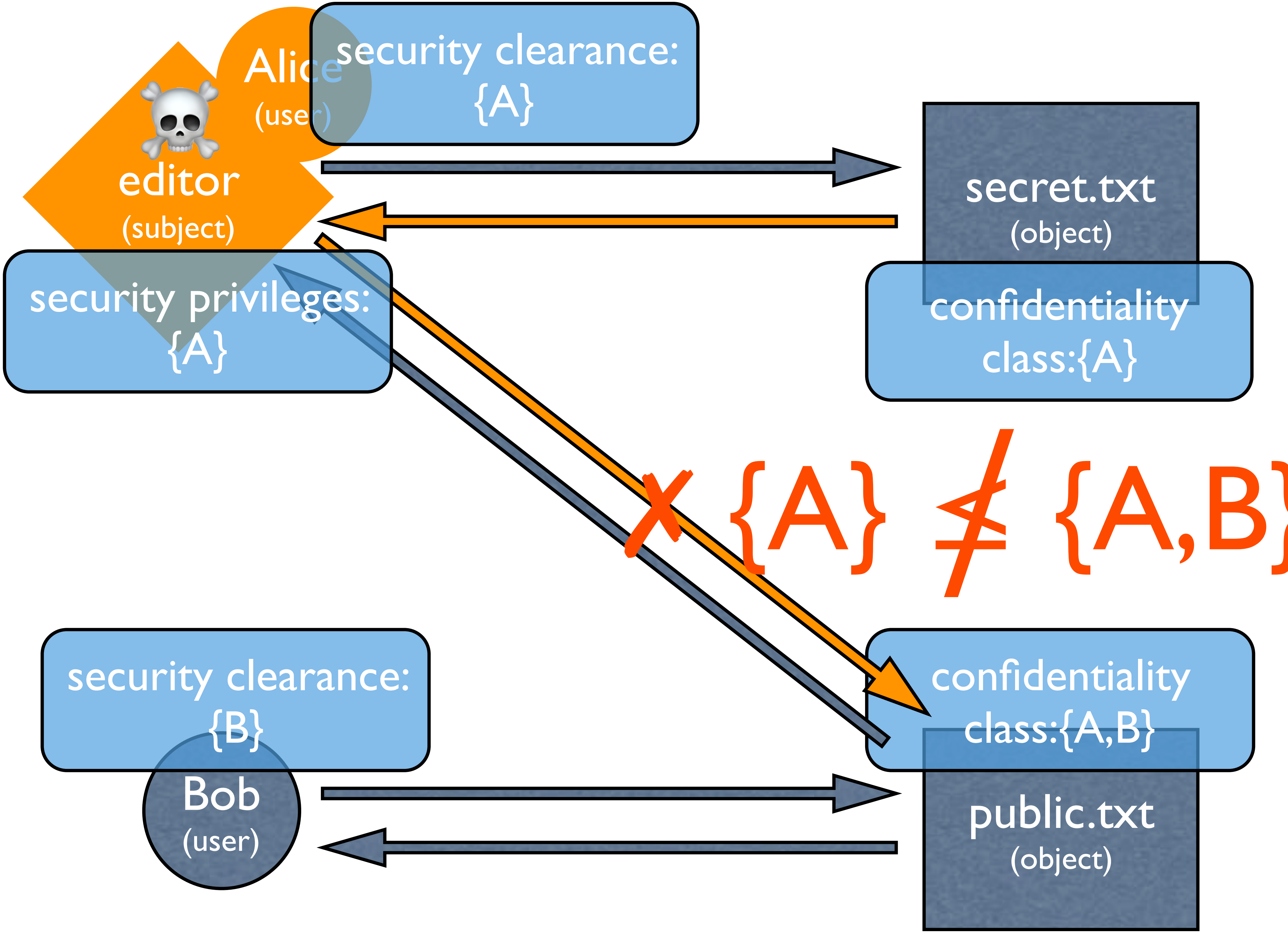


- <u>Confidentiality</u> labels can be compared and are assigned to subjects (s) and objects (o)
- BLP mandatory access rules
  - objects can only be read by subjects with the same or higher confidentiality clearance
  - subjects can only write to objects with the same of higher confidentiality level





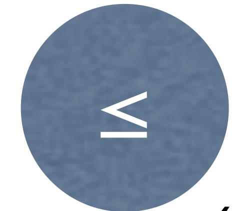
Alice
(user)
security clearance:
{A}
editor
(subject)
security privileges:
{A,B}
secret.txt
(object)
confidentiality
class:{A}
{A} ⊀ {A,B}
confidentiality
class:{A,B}
public.txt
(object)
security clearance:
{B}
Bob
(user)



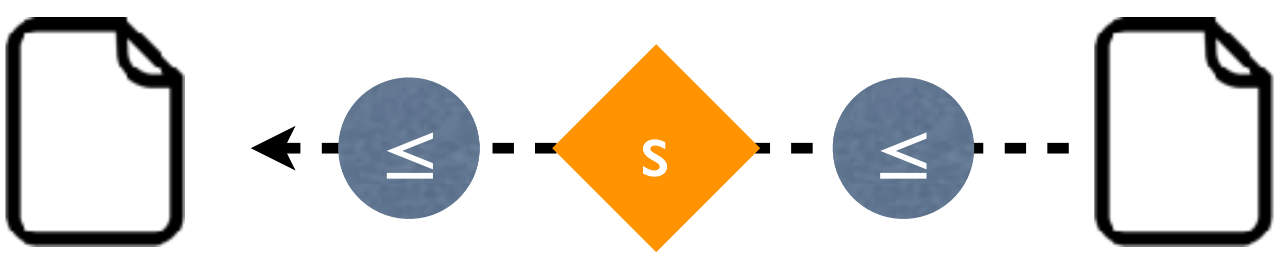
Alice
(user)
security clearance:
{A}
editor
(subject)
security privileges:
{A}
secret.txt
(object)
confidentiality
class:{A}
✗ {A} ⊈ {A,B}
security clearance:
{B}
Bob
(user)
confidentiality
class:{A,B}
public.txt
(object)

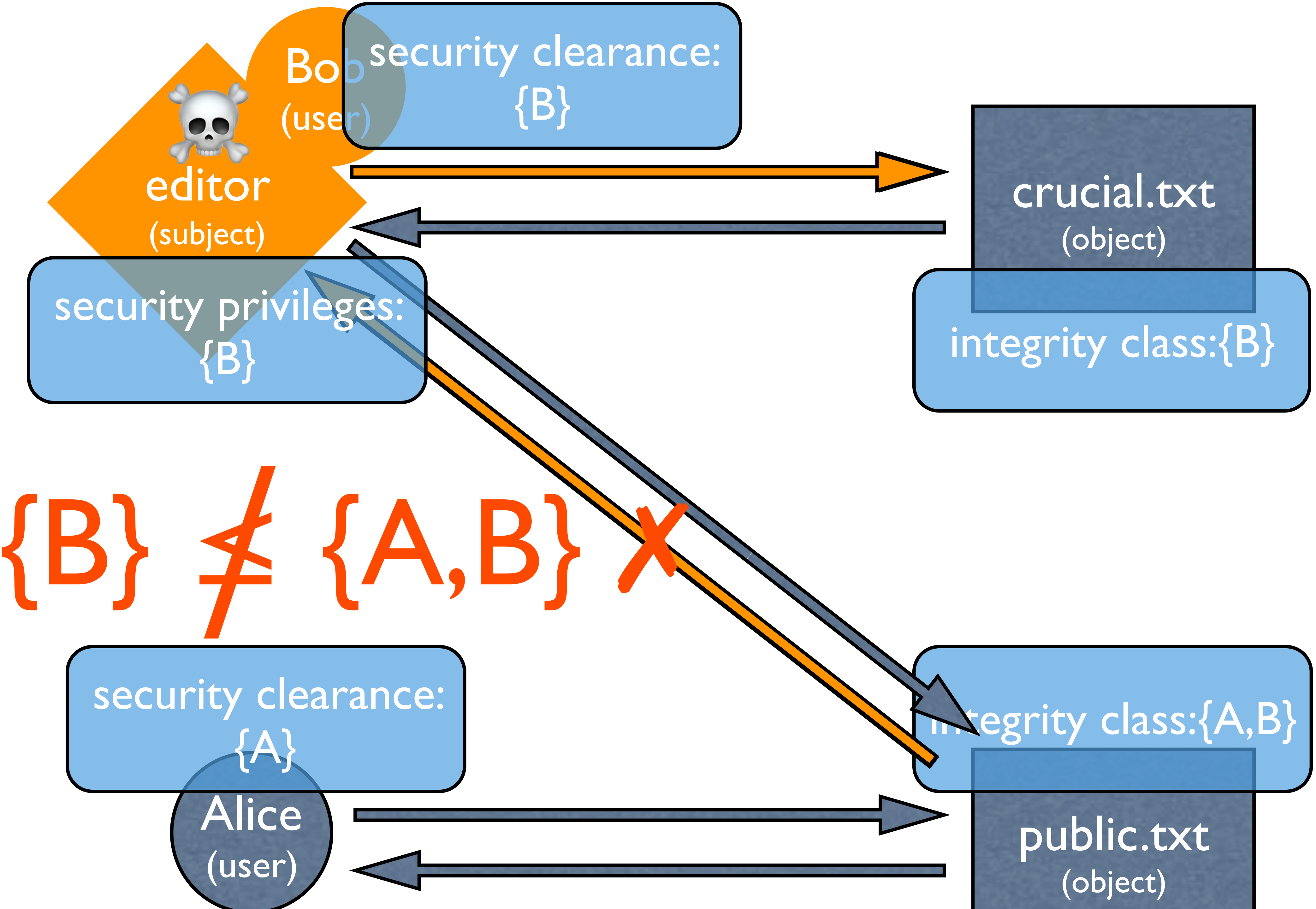
[Biba, 1977]

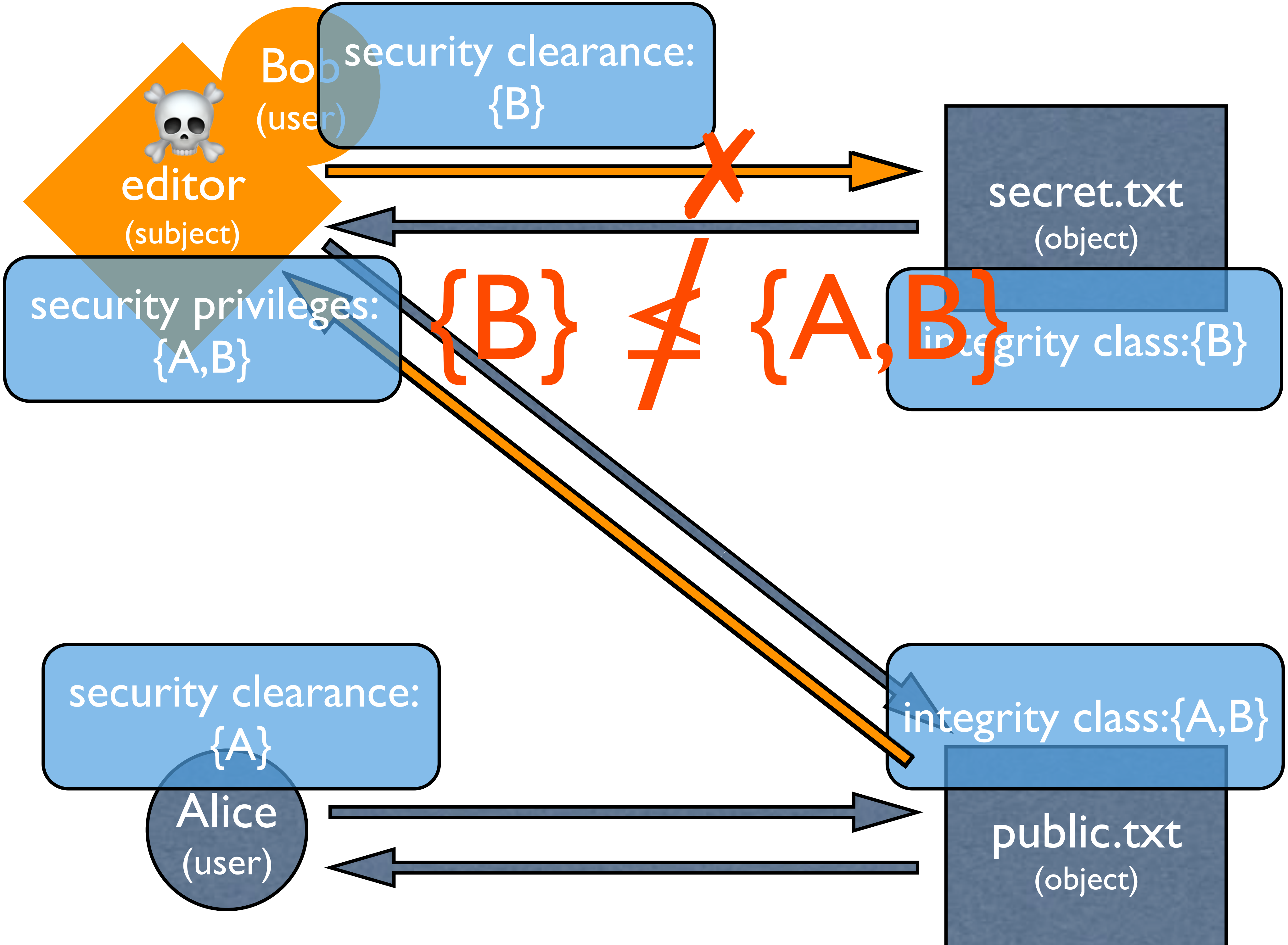
Example: Biba



- Integrity labels can be compared and are assigned to subjects (s) and objects (o)
- Biba mandatory access rules
  - subjects can only read objects with the same of higher integrity level
  - objects can only be written by subjects with the same or higher integrity clearance

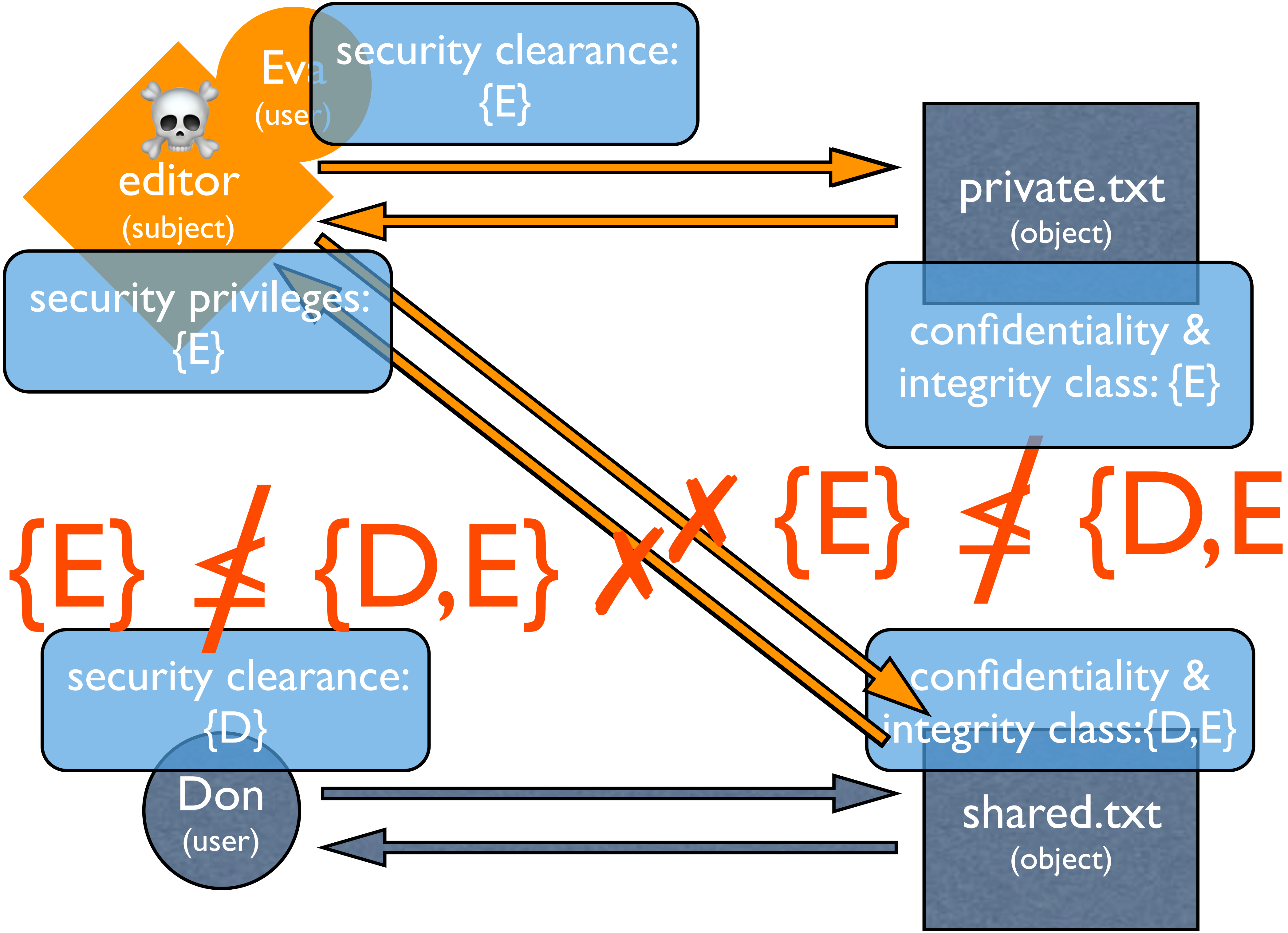



Bob
(user)
security clearance:
{B}
editor
(subject)
security privileges:
{B}
crucial.txt
(object)
integrity class:{B}
{B} ≰ {A,B} ✗
security clearance:
{A}
Alice
(user)
integrity class:{A,B}
public.txt
(object)

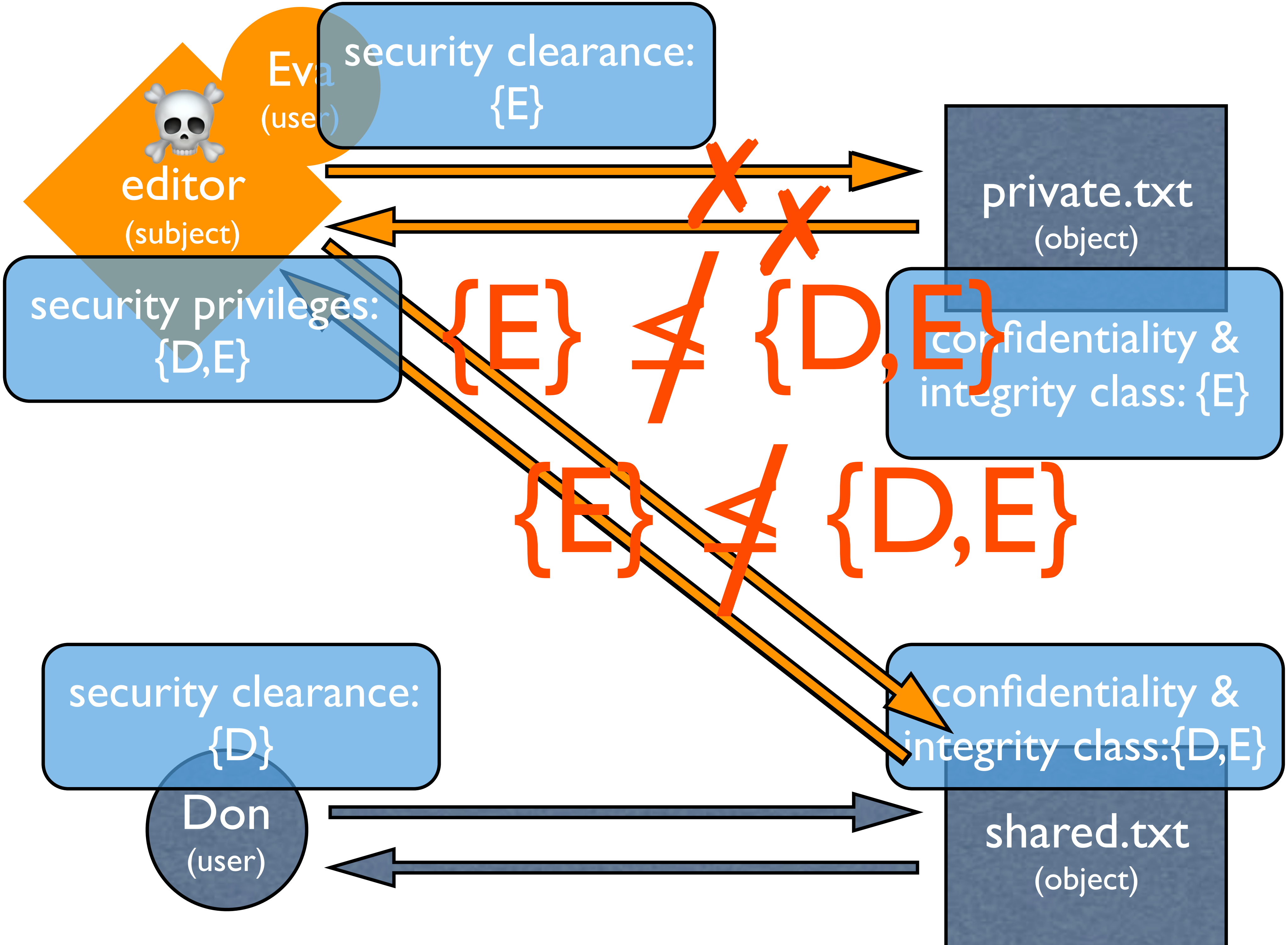

Bob
(user)
security clearance:
{B}
editor
(subject)
secret.txt
(object)
security privileges:
{A,B}
{B} ⊉ {A,B}
integrity class:{B}
security clearance:
{A}
Alice
(user)
integrity class:{A,B}
public.txt
(object)

Confidentiality + Integrity: a composite model

- Confidentiality and integrity labels form a lattice (product of a confidentiality and an integrity lattice)
- λ and ω assign confidentiality and integrity labels to subjects (s) and objects (o)
- Mandatory access rules:
  - simple security: s can read o only if $\lambda(o) \leq \lambda(s)$ and $\omega(s) \leq \omega(o)$
  - ★-security: s can write to o only if and $\lambda(s) \leq \lambda(o)$ and $\omega(o) \leq \omega(s)$



Eva
(user)
security clearance:
{E}
editor
(subject)
security privileges:
{E}
private.txt
(object)
confidentiality &
integrity class: {E}
{E} ⋠ {D,E}
✗
✗
{E} ⋠ {D,E
security clearance:
{D}
Don
(user)
confidentiality &
integrity class:{D,E}
shared.txt
(object)



Eva
(user)
security clearance:
{E}
editor
(subject)
security privileges:
{D,E}
private.txt
(object)
confidentiality &
integrity class: {E}
{E} ⋠ {D,E}
{E} ⋠ {D,E}
security clearance:
{D}
Don
(user)
confidentiality &
integrity class:{D,E}
shared.txt
(object)

Example

Authority of the process	secret.txt	crucial.txt	public.txt
Alice	r	-	r
Bob	-	w	w

- Now can you conceive a program that discloses Alice's secret information?
  Or ruins Bob's crucial information?
- What are the disadvantages of these restrictions?

Covert information flow

- Consider two files:
  - "secret.txt" - owned by Alice, can only be read and written by Alice
  - "public.txt" - can be read and written by everyone
- Can you conceive a program that discloses the secret information to an attacker that can see non-termination?
  - Can Discretionary Access Control prevent it?
  - Can Mandatory Access Control prevent it?

From Access Control...

- In the absence of knowledge about what a given program does, decisions are necessarily coarse to ensure confidentiality and integrity.
  - Discretionary access control cannot ensure end-to-end policies.
  - Mandatory access control can do so using very restrictive policies.

… to Information Flow Control

"By decoupling the right to access information from the right to disseminate it, the flow model goes beyond the access matrix model in its ability to specify secure information flow. A practical system needs both access and flow control to satisfy all security requirements."

D. Denning, 1976

- Information flow control focuses on how information is allowed to flow once an access control is granted.

Class Outline

- Information Flow Security
  - introduction
  - Access Control to Information Flow Control
  - Encoding and exploiting information flows

Encoding and exploiting information flows

☞ "Language-Based Information-Flow Security", A. Sabelfeld and A. Myers , 2002.

“no illegal flows” property

- We want to ensure that propagation of information by programs respects information flow policies, i.e. there are no illegal flows
- “no illegal flows” = an attacker cannot infer secret input or affect critical output by inserting inputs into the system and observing its outputs.
- How can we express the property of whether a program respects information flow policies?

Suppose YOU are the attacker

- You can only read and write to variables of “low level” -- i.e., your level or lower
- For each of the following programs:
  - Can you use the program to uncover “high level” information?
  - Can you use the program to affect “high level” information?

Following examples

- Language: standard imperative language where information containers are variables.
- Notation: x_L denotes that a variable x has security level L.
- Information flow policy:

Confidentiality

secret
|
you

Integrity

you
|
untainted

Preserves confidentiality?

- $y_{secret} := x_{you}$ ✓
- $x_{you} := y_{secret}$ ✗

Explicit leak

- if y_{secret} then $x_{you} := 0$ else $x_{you} := 0$ ✓
- if y_{secret} then $x_{you} := 0$ else $x_{you} := 1$ ✗
- if x_{you} then $y_{secret} := 0$ else $y_{secret} := 1$ ✓
- $x_{you} := 0$; while $x_{you} < y_{secret}$ do $x_{you} := x_{you} + 1$ ✗

Implicit leak

- $x_{you} := 1$; while y_{secret} do $x_{you} := 0$?
- while x_{you} do skip ; $y_{secret} := 0$ ✓

Depends

Preserves integrity?

- $y_{untainted} := x_{you}$
- $x_{you} := y_{untainted}$
- if $y_{untainted}$ then $x_{you} := 0$ else $x_{you} := 0$
- if $y_{untainted}$ then $x_{you} := 0$ else $x_{you} := 1$
- if x_{you} then $y_{untainted} := 0$ else $y_{untainted} := 1$
- $y_{untainted} := 0$; while $x_{you} > y_{untainted}$ do $y_{untainted} := y_{untainted} + 1$
- $x_{you} := 1$; while y_{secret} do $x_{you} := 0$
- while x_{you} do skip ; $y_{untainted} := 0$?

Explicit leak

Implicit leak

Depends

What is the power of the attacker?

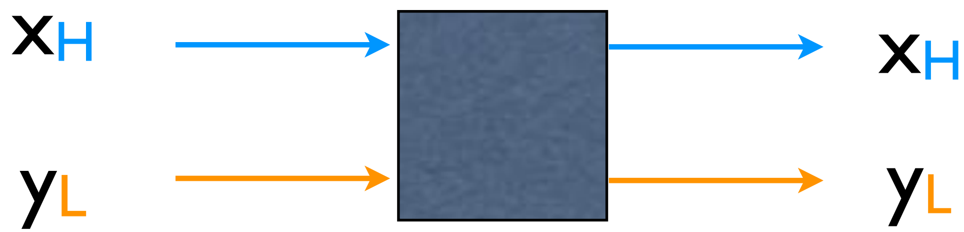
- Can the attacker observe:
  - That a program does not terminate?
  - Intermediate steps of the computation?
  - The possibility of producing certain states?
  - The likelihood of producing certain states?

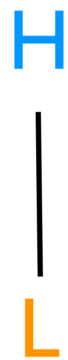
Challenge for next class

- Secure program = the program does not encode illegal information flows
- Can you formulate a security property that defines when a program is secure?

Tips

- Suppose you are the attacker. You are given a black box program, and you can control the low inputs of the program, and read its low outputs.
  - How many executions would you need to find out if the program leaks information?
  - What inputs would you give it in order to find out whether the program leaks information?





- Which of the following experiments allows you to conclude whether the program encodes leaks?
  - Input: y_L=0, Output: y_L=0
  - Input: y_L=0, Output: y_L=1
  - Input: y_L=0, Output: y_L=0 and Input: y_L=0, Output: y_L=0
  - Input: y_L=0, Output: y_L=0 and Input: y_L=1, Output: y_L=1
  - Input: y_L=0, Output: y_L=0 and Input: y_L=0, Output: y_L=1

Conclusion

- Access Control mechanisms do not have the appropriate refinement for enforcing Information Flow policies, as they do not take into account program behaviors.
- In order to analyze information flows in a given program, it is necessary to consider its possible behaviors.
- Next class: Definition of Security Properties