



# Maintaining Privacy on Derived Objects

N. Zannone<sup>a b</sup> and S. Jajodia<sup>b</sup> and F. Massacci<sup>a</sup> and D. Wijesekera<sup>b</sup>

<sup>a</sup> Dep. of Information and Communication Technology, University of Trento

<sup>b</sup> Center for Secure Information Systems, George Mason University



# Summary

---

- Access Control & Privacy
- Access Control Policies and User Preferences
- Information Flow Control
- Creating objects
  - Conditions for creating objects
  - Authorizations on derived objects
- Derivation Tree
- Conclusion and future work

# Access Control

- Essential for building secure information systems
- Protect the confidentiality of information
- An authorization is a triple of the form  $(o, s, \langle sign \rangle a)$ 
  - $(o, s, +a)$ : subject  $s$  is authorized to execute action  $a$  on object  $o$
  - $(o, s, -a)$ : subject  $s$  is denied to execute action  $a$  on object  $o$
- Authorization frameworks manage access to data by users
  - For any access request, exactly one decision (allowed/denied) is provided



# Privacy

---

“Privacy is the right of individuals to determine for themselves when, how, and to what extent information about them is communicated to others”

*Alan Westin*<sup>a</sup>

- Data owners directly specify their preferences
  - Who can access their information
  - How it can be used

---

<sup>a</sup>Professor of Public Law and Government at Columbia University

# Access Control Policies

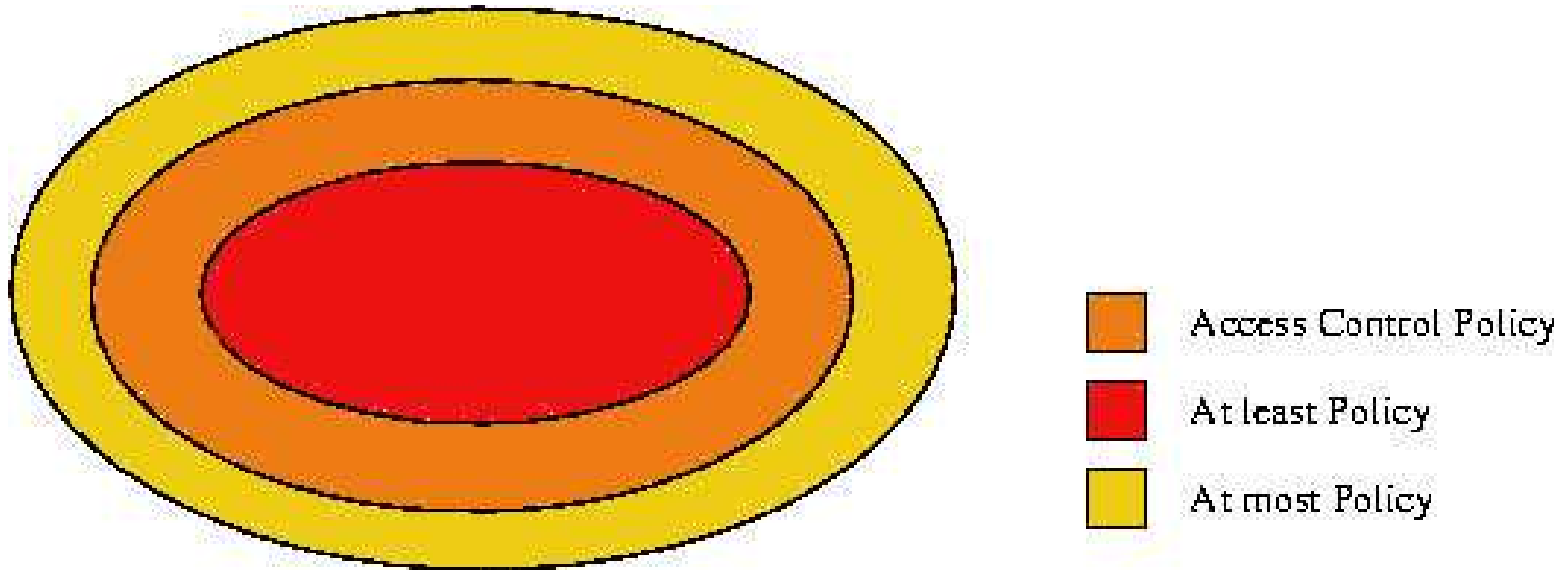
- Determine which entities are entitled to access an object and which actions they can perform on it
- Defined by the system administrator in agreement with enterprise policies
- An *access control policy* is a set of positive authorizations
  - $policy(o) = \{(s, a) \mid (o, s, a) \in AUTH^+\}$
  - $policy(o)$  returns the access control list associated with  $o$

# User Preferences

- A data owner may want to maintain permissions on his objects to check that they are not misused
- A data owner may want to restrict authorizations on his objects
- These represent user preferences and can be modeled through two sets of authorizations
  - At least policy
    - $policy_{\leq}(o)$  returns the authorizations that  $o$  should have
  - At most policy
    - $policy_{\geq}(o)$  returns the authorizations that  $o$  at most can have

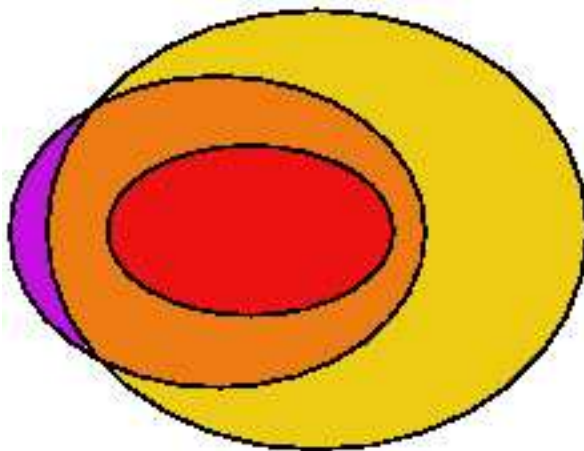
# Access Control Policy vs User Preferences

- User preferences represent the range in which authorizations can be granted
  - $policy_{\geq}(o) \subseteq policy(o) \subseteq policy_{\leq}(o)$

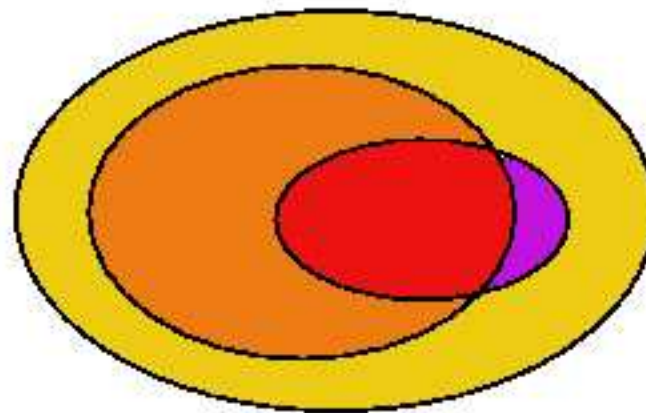


# Zombie Objects

- Conflicts can arise between enterprise policies and user preferences
  - *Zombie objects*: access control does not comply with user preferences
  - Every access to zombie objects is blocked until conflicts are resolved



$$policy(o) \not\subseteq policy_{\leq}(o)$$



$$policy_{\geq}(o) \not\subseteq policy(o)$$



# Information Flow Control

---

- Information systems manipulate information
  - The outcome of a data processing can be seen as a new object
  - Derived objects contain information belonging to the objects used to derive it
  - Information systems may release information as part of their functionalities
- Need to introduce information flow control
  - Ensure that information are not disclosed to unauthorized entities

# Creating objects

- Information systems support data processing for manipulating information
- Represent data processing as function
  - e. g.,  $f(s, o_1, \dots, o_m) = o$
- For enforcing data protection, we should answer
  - Is the subject  $s$  entitled to create the derived object  $o$ ?
  - Who is authorized to access the derived object  $o$ ?

# Conditions for Creating Objects

- Subjects may need to use existing objects
- Only users that play a certain role or belong to a certain group may be entitled to create the object
- Make explicit the conditions under which a subject can create an object

$$\blacksquare f(s, o_1, \dots, o_m) = \begin{cases} o & \text{if } \mathcal{C} \text{ is true} \\ \perp & \text{otherwise} \end{cases}$$

- $\mathcal{C}$  represents the condition that must be satisfied
- $\perp$  means that object  $o$  cannot be created

# Authorizations on Derived Objects

- Once an object is created, we should associate with the object
  - Access control policy
  - User preferences
- The derived object is not independent from the objects used to derived it
  - The policies should take into account the authorizations associated with the objects used to derive it
- Not all data processing disclose information
- Taxonomy of functions
  - *disclosure functions*
  - *non disclosure functions*

# Disclosure Functions (I)

- Derived objects disclose information about the objects used to create it
- The policy associated with the object is the intersection of the policies associated with the objects used to derive it
- Some information must be disclosed for satisfying availability requirements
  - Privacy Act allows an agency to disclose data without the consent of the data owner to those officers and employees of the agency who need the data to perform their duties
- Some accesses should be restricted
  - A bank does not consider “reasonable” that a client modifies his account balance by himself

# Disclosure Functions (II)

## ■ Access control policy

- $policy(f_{DF}(s, o_1, \dots, o_m)) = \left( \left( \bigcap_{i \in [1, \dots, m]} policy(o_i) \right) \cup \mathcal{P}_1 \right) \setminus \mathcal{P}_2$

- $\mathcal{P}_1$  is the policy used to grant access for guaranteeing availability requirements

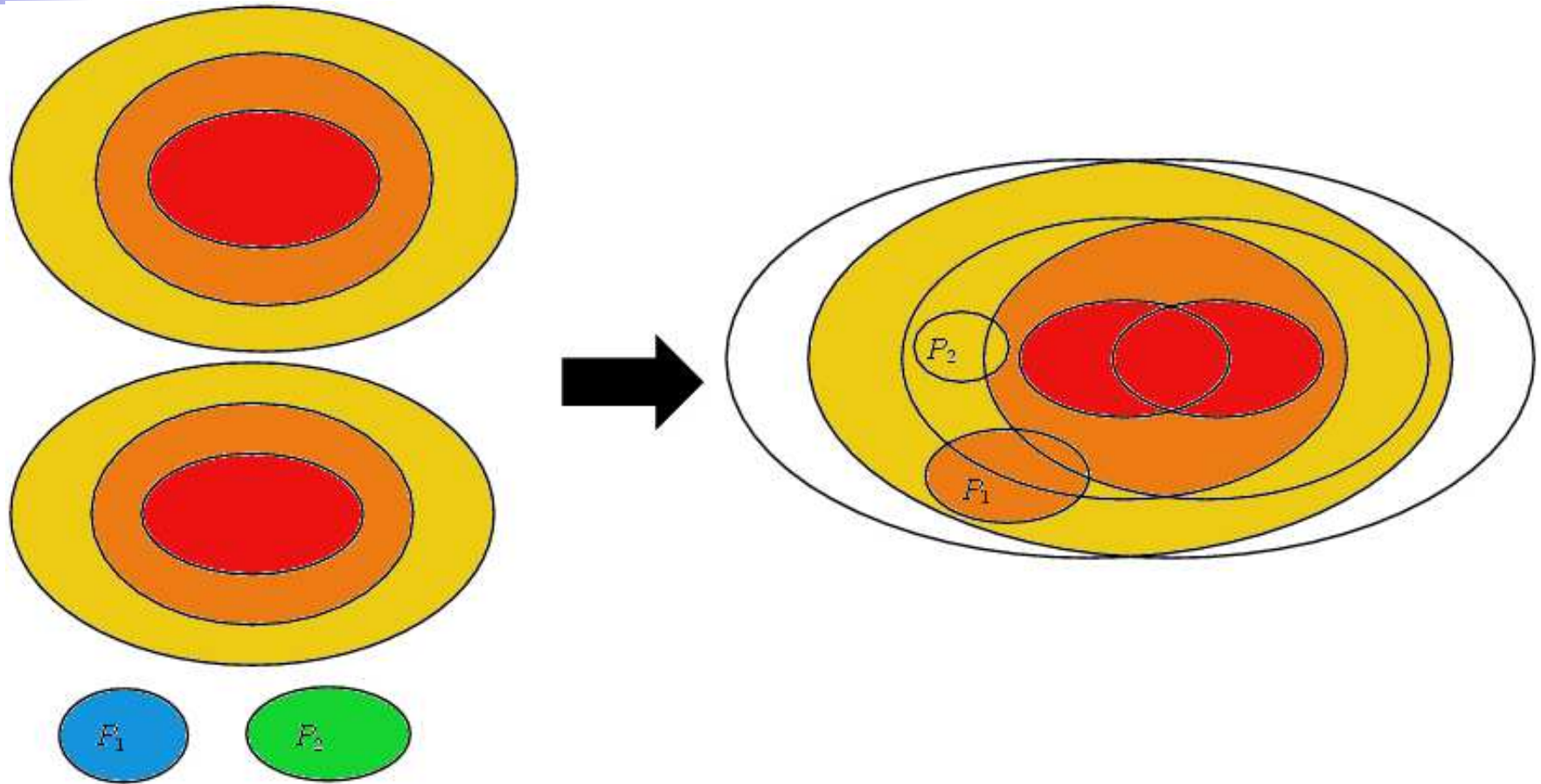
- $\mathcal{P}_2$  is the policy used to limit the access to the object

## ■ User preferences

- $policy_{\geq}(f_{DF}(s, o_1, \dots, o_m)) = \bigcup_{i \in [1, \dots, m]} policy_{\geq}(o_i)$

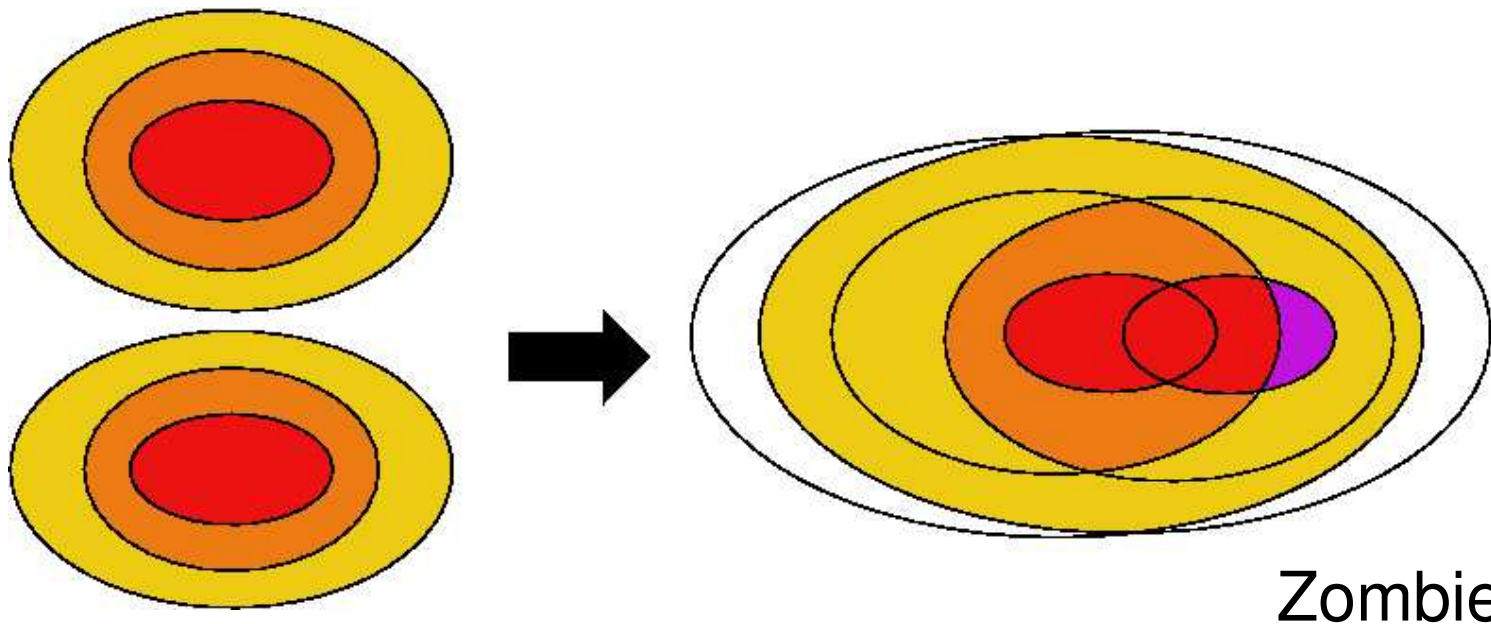
- $policy_{\leq}(f_{DF}(s, o_1, \dots, o_m)) = \bigcap_{i \in [1, \dots, m]} policy_{\leq}(o_i)$

# Disclosure Functions (III)



# Conditions

- Ensure that the derived object is not a zombie objects
  - Access control policy associated with the derived object has to be compared with user preferences
    - $\forall j, i \in [1, \dots, m] \text{ policy}_{\geq}(o_j) \subseteq \text{policy}(o_i)$





# Non Disclosure Function (I)

---

- Functions such as statistical operations do not disclose sensitive information
- The disclosure of information is not sufficient to trace the origin of the information itself
- Policies can be “relaxed”
  - Privacy Act does not impose any conditions on aggregate statistical data without any personal identifiers

# Non Disclosure Function (II)

## ■ Access control policy

- $policy(f_{NDF}(s, o_1, \dots, o_m)) = \left( \left( \bigcup_{i \in [1, \dots, m]} policy(o_i) \right) \cup \mathcal{P}_1 \right) \setminus \mathcal{P}_2$

- $\mathcal{P}_1$  is the policy used to grant access for guaranteeing availability requirements

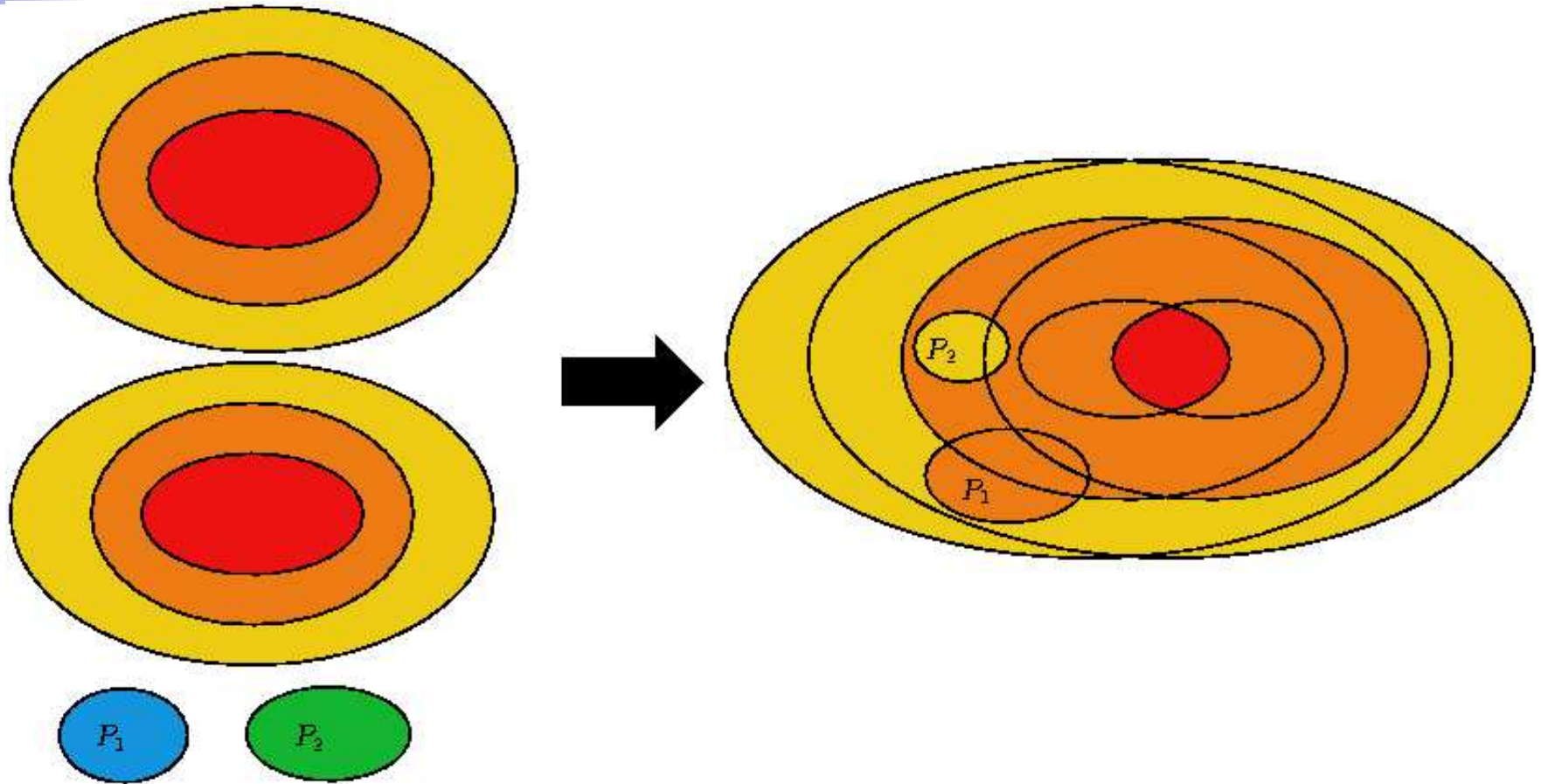
- $\mathcal{P}_2$  is the policy used to limit the access to the object

## ■ User preferences

- $policy_{\geq}(f_{NDF}(s, o_1, \dots, o_m)) = \bigcap_{i \in [1, \dots, m]} policy_{\geq}(o_i)$

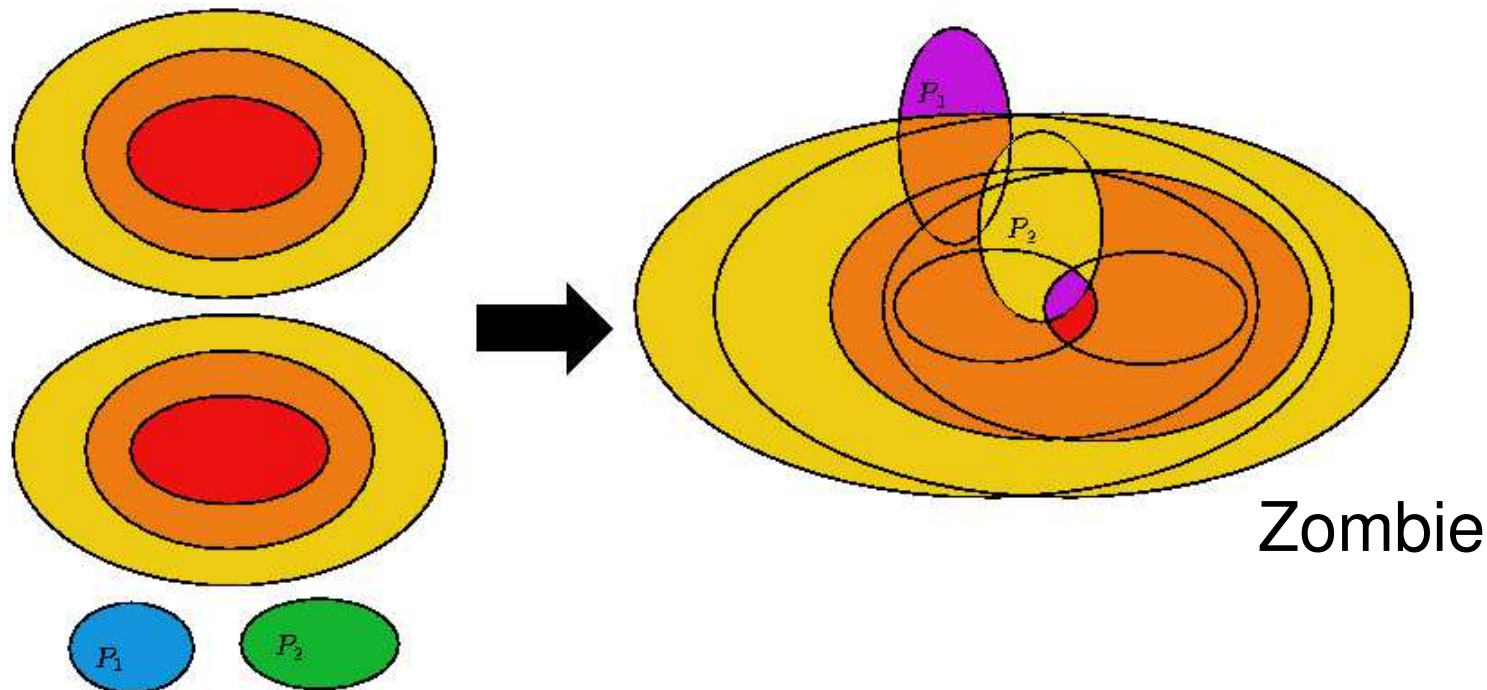
- $policy_{\leq}(f_{NDF}(s, o_1, \dots, o_m)) = \bigcup_{i \in [1, \dots, m]} policy_{\leq}(o_i)$

# Non Disclosure Functions (III)



# Conditions

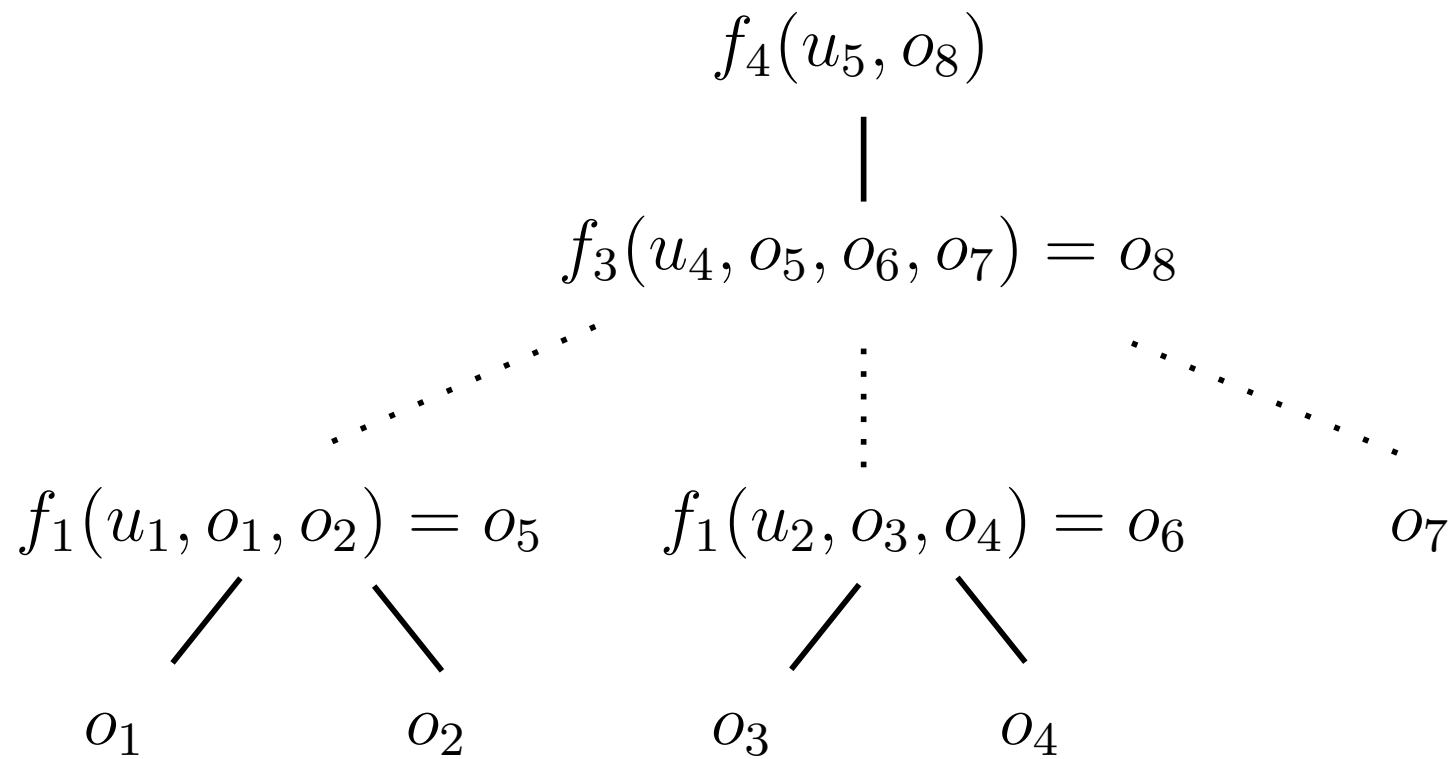
- Ensure that the derived object is not a zombie object
  - Access control policy has to be compared with user preferences
    - $\bigcap_{i \in [1, \dots, m]} policy_{\geq}(o_i) \cap \mathcal{P}_2 = \emptyset$
    - $\mathcal{P}_1 \setminus \mathcal{P}_2 \subseteq \bigcup_{i \in [1, \dots, m]} policy_{\leq}(o_i)$



# Derivation Trees

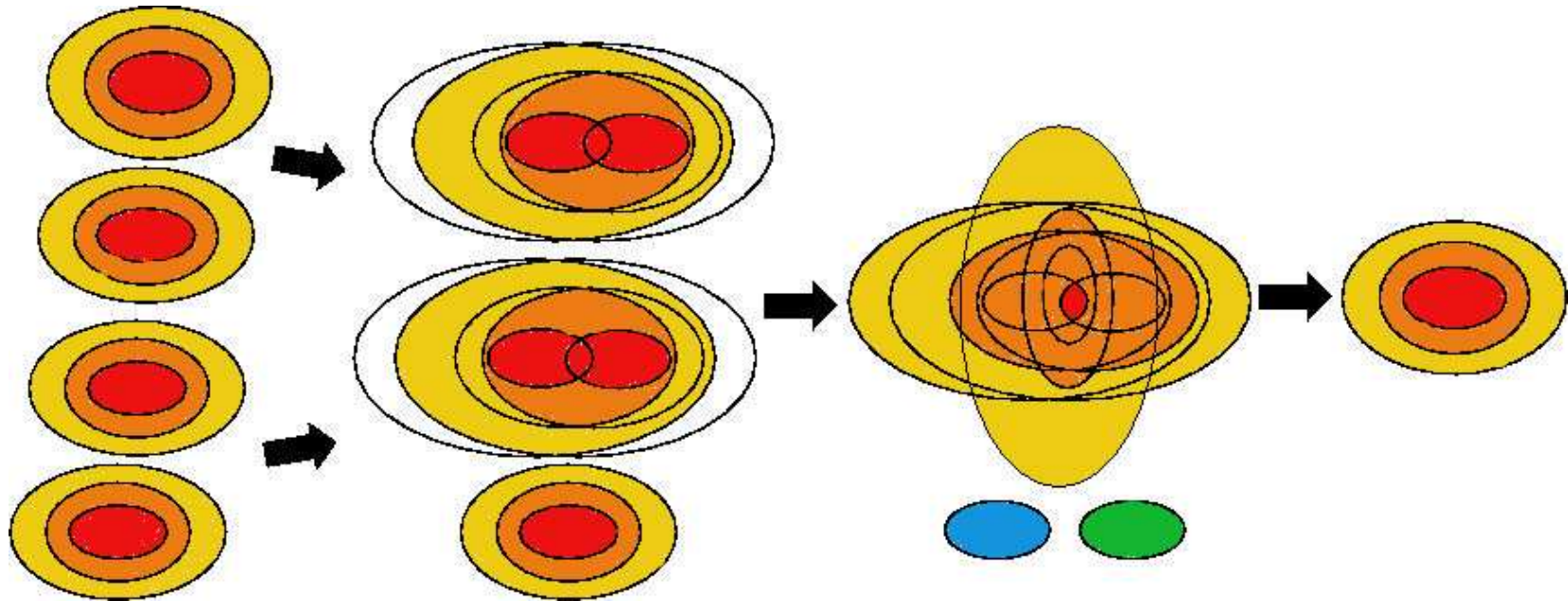
- The outcome of a data processing may be used as input for other data processing
- The process to derive an object can be seen as a tree
  - Root is the derived object
  - Leaves are primitive objects (i.e. objects not derived by using functions)
  - Edges
    - Disclosure step (full edge)
    - Non disclosure step (dotted edge)

# Example



# Guaranteeing Data Protection

- Verify the entire process used to derive the object



# Conclusions

- Verify permissions for creating objects
- Automatically derive access control policies for derived objects
- Enforce access control policies taking into account user preferences
- Future works
  - Extend the notion of access control policy and user preferences in order to take into account negative authorizations
  - Define mechanisms in order to solve possible conflicts
  - Formalize in FAF the process for enforcing access control policies concerning derived objects