

IV PET for Protecting Personal Data

- Security Models enforcing legal privacy requirements
→ A Formal Task-Based Privacy Model
- Cryptography , Steganography

11. A Formal Task-Based Privacy Model

11.1 Basic Privacy Requirements:

- necessity of data collection and processing
- purpose specification and binding
(as there are no "non-sensitive" data)
- adequate organisational and technical safeguards

11.2 Privacy Analysis of Security Models

a.) Bell LaPadula-Model /DAC:

b.) RBAC-Model :

Confidentiality:

MAC: is based on sensitivity-levels, but:
Personal data cannot accurately be classified by its sensitivity, as the sensitivity is related to the purpose of use

+

DAC: is based on the "owner"-principle, but:
Personal data about a data subject should not be owned by any other user

Integrity:

+

Necessity of data processing:

Roles should be only authorised for privileges that are necessary to perform their duties

Purpose binding:

The Bell LaPadula Model (Mitre, 1973):

$\forall S_i$: Subject, O_j, O_k : Object :

Simple Security Property (no read up):

If S_i has read-access to Object O_j

\Rightarrow

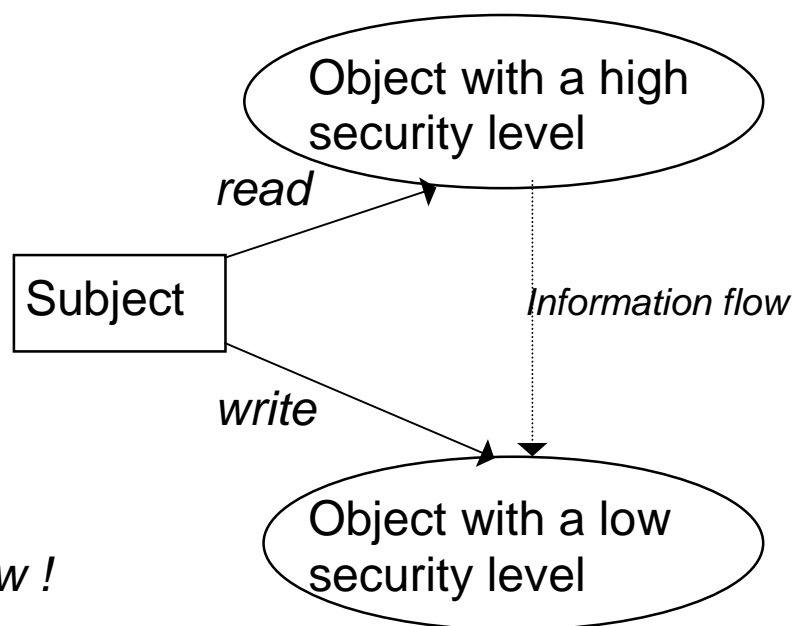
security-level (S_i) \geq security-level (O_j)

*-Property (no write down):

If S_i has read-access to O_j and write-access to O_k

\Rightarrow

security-level (O_k) \geq security-level (O_j)



*The *-property shall prevent such kind of illegal information flow !*

Discretionary Property:

If S_i has x-access to O_j

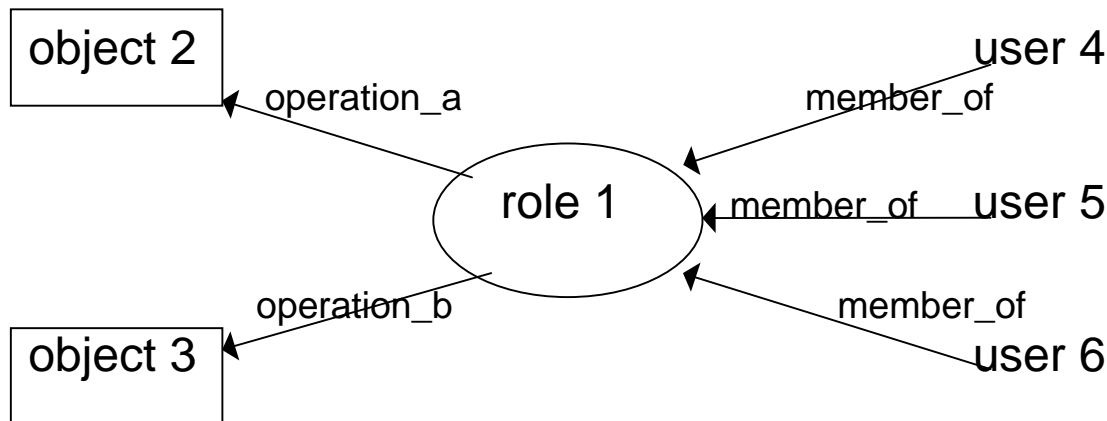
\Rightarrow

$x \in$ Access-Matrix (i,j)

Role-Based Access Control (RBAC), (Ferraiolo/Kuhn, 1992):

In compliance with organisation-specific security guidelines the security administrator determines:

- membership of users to roles
user --> roles
- allocation of privileges (operations) to a role
role --> set of operations



Example:

A user in the role “doctor” may perform the following transactions (operations):

- enter_diagnosis
- prescribe_medication

RBAC- Security Properties:

(Role Execution):

$\forall s: \text{subject}, op: \text{operation}: \text{exec}(s,op) \Rightarrow \text{active-role}(s) \neq \emptyset$

(Role Authorisation):

$\forall s: \text{subject}, op: \text{operation}: \text{active-role}(s) \subseteq \text{authorised-role}(s)$

(Operation Authorisation):

$\forall s: \text{subject}, op: \text{operation} \exists r: \text{roles}: \text{exec}(s,op) \Rightarrow$
 $r \in \text{active-roles}(s) \wedge op \in \text{role-operations}(s)$

(Object Access Authorisation):

$\forall s: \text{subject}, o: \text{object}: \text{access}(s,o) \Rightarrow \exists r: \text{roles}, op: \text{operation}:$
 $r \in \text{active-roles}(s) \wedge op \in \text{role-operations}(r) \wedge$
 $o \in \text{operation-object}(op)$

(Role Hierarchy):

$\forall s: \text{subject}, r_i, r_j: \text{roles}: r_j \in \text{authorised-roles}(s) \wedge r_j > r_i \Rightarrow$
 $r_i \in \text{authorised-roles}(s).$

(Static Separation of Duty):

$\forall u: \text{user}, r_i, r_j: \text{roles}: i \neq j: u \in \text{role-members}(r_i) \wedge$
 $u \in \text{role-members}(r_j) \Rightarrow r_i \notin \text{mutually-exclusive-authorisation}(r_j)$

(Dynamic Separation of Duty):

$\forall s: \text{subject}, r_i, r_j: \text{roles}. i \neq j: r_i \in \text{active-roles}(s) \wedge$
 $r_j \in \text{active-roles}(s) \Rightarrow r_i \notin \text{mutually-exclusive-activation}(r_j)$

11.3 A Formal Task-based Privacy Model:

$\forall S$: Subjects, O : Personal Data Objects :

Task-Authorisation:

current-task (S) \in authorised-tasks (S)

TP-Authorisation:

Current-TP (S) \in authorised-TP (current-task (S))

Necessity of data processing:

If S has x-access to O

\Rightarrow

(current-task (S), class (O), current-TP (S), x) \in
necessary-accesses

Purpose binding:

If S has x-access to O

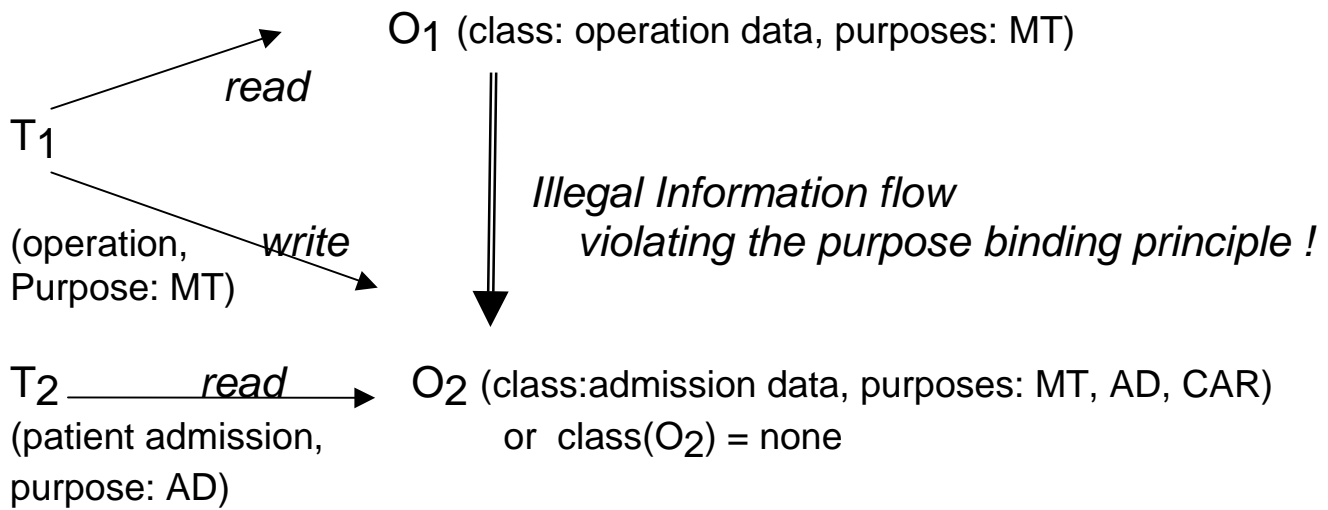
\Rightarrow

purpose (current-task (S)) \in purposes (class (O))

\vee

(purpose (current-task (S)), O) \in consent

Information flow Control:



MT: medical treatment, AD: administration, CAR: intensive care

The following condition has to be guaranteed:

$\forall S_i$: Subject, O₁, O₂: Object :

If S_i has read-access to O₁ and write- or append-access to O₂
 \Rightarrow Purposes (class (O₁)) \supseteq Purposes (class (O₂)).

Flow-Secure Access Control:

New state variables:

Input-Purposes, Output-Purposes: $S \rightarrow 2^P$

Input-Purposes(S_i) = set of purposes that the classes of all objects, which S_i has read, have in common.

Initially: Input-Purpose (S_i) = P .

If S_i gets read-access to an object O_j :

Input-Purposes(S_i)*=

Input-Purposes(S_i) \cap Purposes(class (O_j)).

Output-Purposes(S_i) = union set of purposes of the classes of objects, to which S_i has been granted write- or append- access.

Initially: Output-Purpose (S_i) = \emptyset .

If S_i gets write- or append-access to an object O_j :

Output-Purposes(S_i)*=

Output-Purposes(S_i) \cup Purposes(class (O_j)).

Information flow control property:

$\forall S_i$: Subject:

Output-Purposes (S_i) \subseteq Input-Purposes (S_i)

State Transition Functions:

get-access

create- / delete-object

change-current-task

execute-TP, exit-TP

get-write-access (S_i, O_j, x)

(Semantics: Subject S_i requests that access to object O_j in usage mode x be enabled, $x \in \{\text{write, append}\}$)

If [($O_j \in OP$)

and

(Current-Task (S_i), Class (O_j), Current-TP(S_i), x) \in NA

and

[((Purpose (Current-Task (S_i)) \in Purposes (Class (O_j))

or

(Purpose (Current-Task (S_i)), O_j) \in Consent)]

and

Output-Purposes (S_i) \cup Purposes (class (O_j))

\subseteq Input-Purposes (S_i)]

or

[($O_j \notin OP$)

and

Purposes (class (O_j)) \subseteq Input-Purposes (S_i)]

(*or Input-Purposes (S_i) = P^*)

then

$CA^* = CA \cup \{(S_i, O_j, x)\}$

Output-Purposes * (S_i) = Output-Purposes (S_i)

\cup Purposes (class (O_j))

Privileged functions (to be performed by data protection officer in cooperation with security officer):

- add- / delete-NA, add- / delete-object-class
- add- / delete-task, add- / delete-authorized-TP
- add- / delete-purpose, add- / delete-consent
- add- / delete-responsible-users

Data Protection Officer S_j issues ticket:

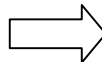
Ticket TKT_j
issued by S_j

Function-type:
add-NA
Parameter-list:
 $T_k, o-class_m, transp_n, x$

Security Officer S_i can use this ticket to perform a privileged function:

add-NA (S_i, TKT_j (*function-type, $T_k, o-class_m, transp_n, x$*))

If role (S_i) = sec-officer



and

function-type = add-NA

and

role (S_i) = data-protection-officer

then

$NA^* = NA \cup$

$\{ (T_k, o-class_m, transp_n, x) \}$

$TKT^* = TKT - \{ TKT_j (function, T_k, o-class_m, transp_n, x) \}$.

Privileged function (to be performed by "responsible user" in cooperation with security officer):

- add- / delete-authorized-task

Privileged function (to be performed by TP-Manager):

- create- / delete-TP

Example: Hospital information system

Purposes: MT (medical treatment), AD (administration), CAR (care), research (RE)

Tasks: diagnosing (MT) , operation (MT), therapy (MT), intensive care (CAR) ,
accounting (AD), statistical analysis (RE),...

Object classes: admission data (AD, MT, CAR), billing data (AD, MT),
diagnosis (MT,CAR), treatment-data (MT), statistics (RE),...

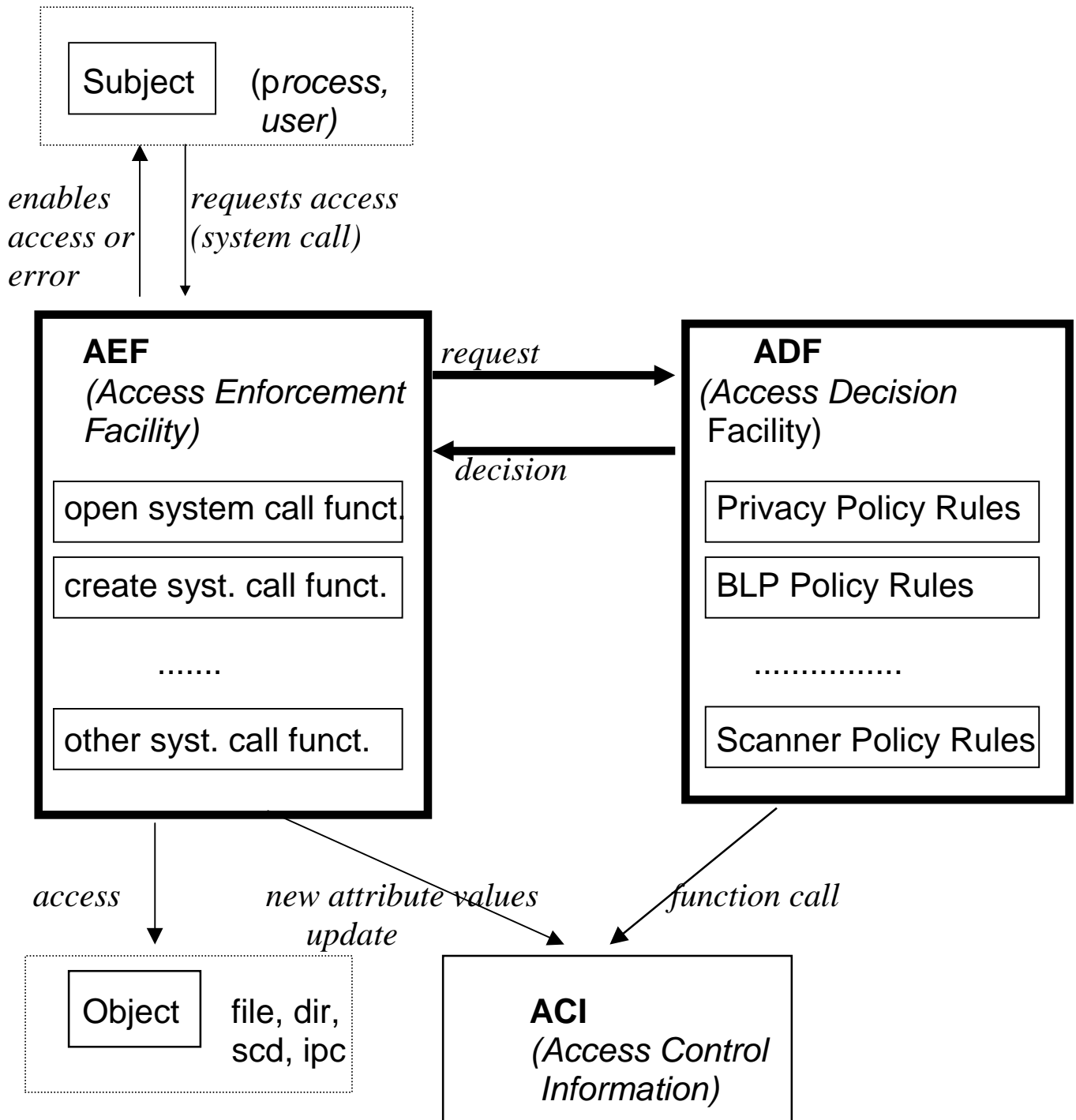
Necessary Accesses: (diagnosing, diagnosis, editor, read/write/create/append)
(diagnosing, billing-data, append-editor, append)
.....
(statistical analysis, diagnosis, statistical program, read)

Task:
Statistical analysis

—————▶ **Diagnosis
of Patient A**

*Access is necessary,
but violates the principle of purpose binding,
unless there is a consent by the data subject !*

11.4 Implementation according to the Generalized Framework for Access Control in Linux



Access Control Information (ACI) for Privacy Policy:

User-ACI	Values
authorised-tasks (-> purpose -> authorised TP -> Responsible) role	a set of task-IDs incl. NIL sec-officer, user, data-protection_officer, tp-manager, system-admin

Process-ACI	Values
owner (pointer to user)	
transformation procedure	a trans.-proc.-ID or NIL
current_task (-> purpose -> authorised-TP -> Responsible)	a task-ID or NIL
process-type	NIL, TP

Object-ACI	Values
class (-> set of purposes)	an object-class-ID or none
transformation-procedure	a trans.-proc.-ID or NIL
object-type	file, dir, ipc, scd
data-type	NIL, TP, personal data, non-personal-data

ADF-Privacy Rule for append-open request:

CASE append-open

SELECT CASE target[input-argument]

CASE file

SELECT CASE data-type(object)

CASE personal-data

IF [Necessary (current-task (process), class (object),
transformation-procedure (process), append)

AND

Purpose-binding (current-task (process),
class (object))

OR

consent (current-task (process), object)]

AND

UNION (output-purposes(process),
purposes(class(object))) in

input-purposes(process)

THEN

return(set-attribute (output-purposes(process),
UNION (output-purposes(process),
purposes(class(object)))); YES)

ELSE

return(NO)

CASE TP (** it is not allowed to directly modify TPs **)

return (NO)

CASE non-personal-data

IF input-purposes(process) is P

THEN

return(set-attribute (output-purposes(process),
P); YES)

ELSE

return(NO)

CASE ELSE

return(NO);

```

CASE ipc
  IF class(object) is none (*object contains non-personal-data*)
  THEN
    IF input-purposes(process) is P
    THEN
      return(set-attribute (output-purposes(process), P);
      YES)
    ELSE
      return(NO)

  ELSE (* object contains personal data *)
    IF Necessary (current-task(process), class(object),
      transformation-procedure (process), append)
    AND
    Purpose-binding (current-task (process), class(object))
    AND
      UNION ( output-purposes(process),
      purposes(class(object))) in
      input-purposes(process)
    THEN (* consents are not defined for ipc-objects *)
      return(set-attribute (output-purposes(process),
      UNION ( output-purposes(process),
      purposes(class(object))))); YES)
    ELSE
      return (NO)

CASE ELSE
  return (UNDEFINED);

```

ACI for Scanner Policy:

Process-ACI	Values
Process-type	MS-trusted or NIL

Object-ACI	Values
Program-type	MS-trusted or NIL
Scan-result	scanned, rejected, unscanned
VS-number	a version number or NIL