Computer Security: Principles and Practice

Chapter 13: Trusted Computing and Multilevel Security

Computer Security Models

- Two fundamental computer security facts
  - All complex software systems have eventually revealed flaws or bugs that need to be fixed
  - It is extraordinarily difficult to build computer hardware/software not vulnerable to security attacks

Confidentiality Policy

- Goal: prevent the unauthorized disclosure of information
  - Deals with information flow
  - Integrity incidental
- Multi-level security models are best-known examples
  - Bell-LaPadula Model basis for many, or most, of these

Formal Security Models

- Problems involved both design and implementation led to development of formal security models
  - Initially funded by US Department of Defense
  - Bell-LaPadula (BLP) model very influential

Bell-LaPadula (BLP) Model

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of security clearance L(s)
- Objects have security classification L(o)

Bell-LaPadula (BLP) Model

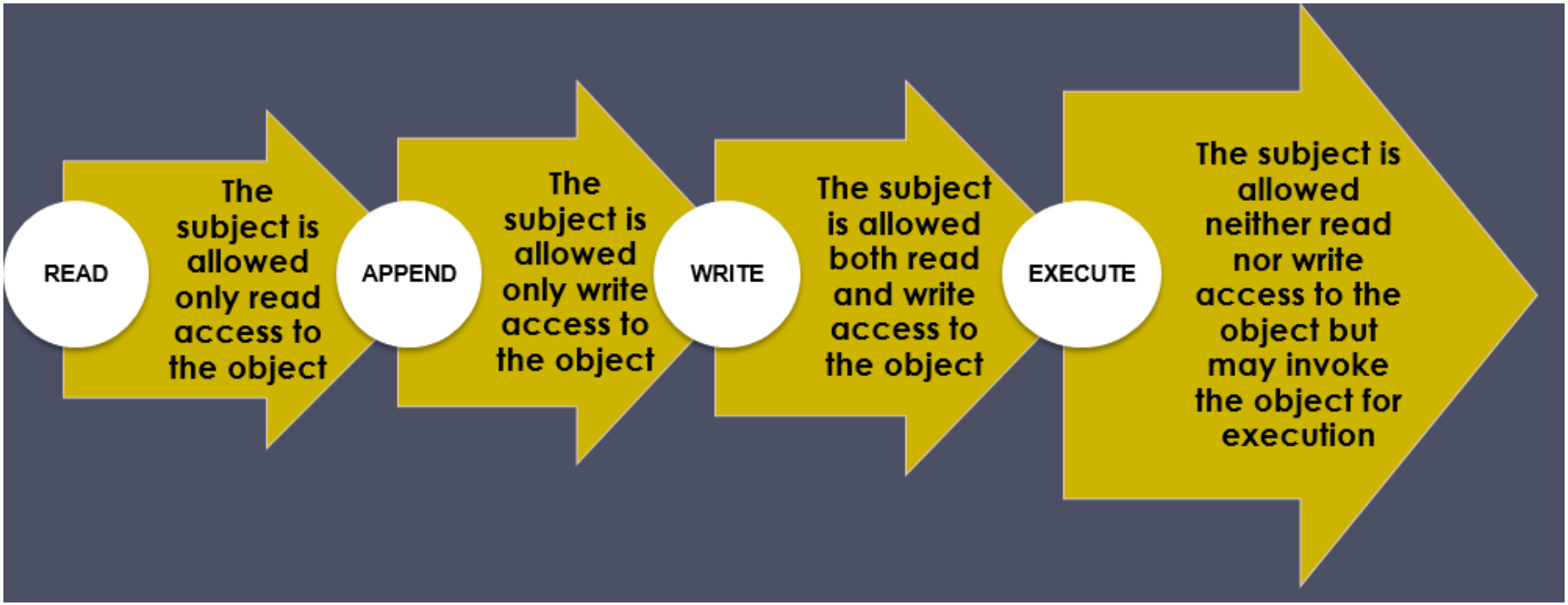
- Developed in 1970s
- Formal model for access control
- Subjects and objects are assigned a security class
- Form a hierarchy and are referred to as security levels
- A subject has a security **clearance**
- An object has a security **classification**
- Security classes control the manner by which a subject may access an object

A BLP Example

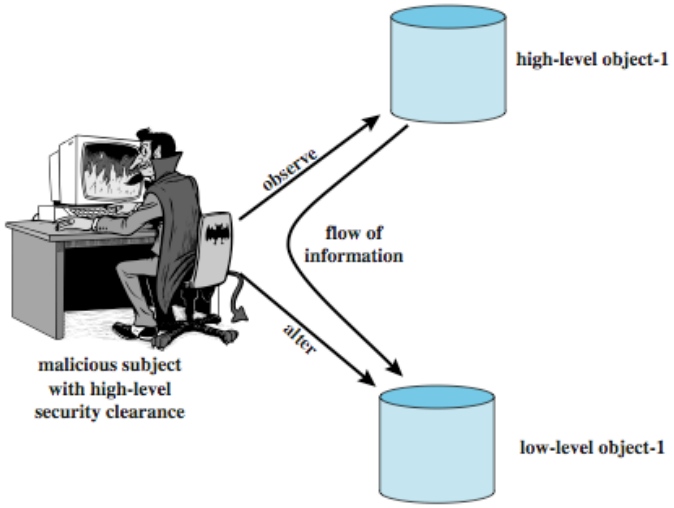
Security level	***Subject***	***Object***
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	James	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- James can only read Telephone Lists

Access Privileges



Multilevel Security



- Multiple levels of security and data
- Subject at a high level may not convey info to a subject at a non-comparable level:
  - No read up (ss-property): a subj can only read an obj of less or equal sec level
  - No write down (*-property): a subj can only write into an obj of greater or equal sec level

BLP Formal Description

- Based on current state of system (*b*, *M*, *f*, *H*):
  - Current access set *b (subj, objs, access-mode);* it is the ***current*** access (not permanent)
  - Access matrix *M* (same as before)
  - Level function *f: assigns sec level to each subj and obj;* a subject may operate at that or lower level
  - Hierarchy *H: a directed tree whose nodes are objs:*
    - *Sec level of an obj must dominate (must be greater than) its parents*

BLP Properties

- Three BLP properties: *(c = current)*
  1. ss-property: (S_i, O_j, read) has $f_c(S_i) \geq f_o(O_j)$
  2. *-property: $(S_i, O_j, \text{append})$ has $f_c(S_i) \leq f_o(O_j)$ and (S_i, O_j, write) has $f_c(S_i) = f_o(O_j)$
  3. ds-property: (S_i, O_j, A_x) implies $A_x \in M[S_i, O_j]$
- BLP give formal theorems
  - Theoretically possible to prove system is secure

BLP Operations

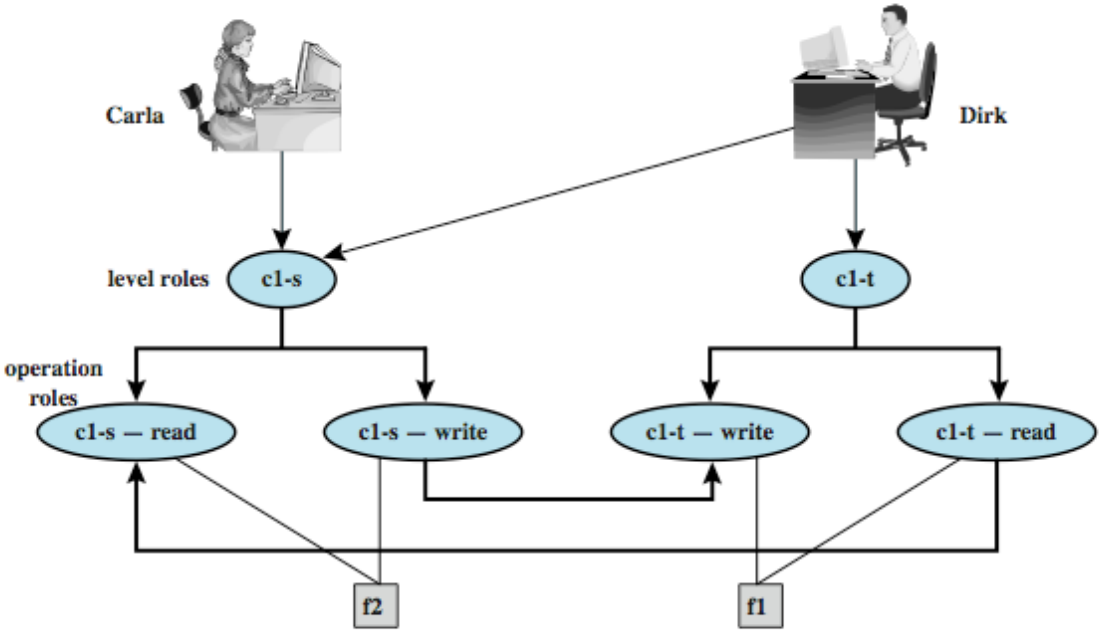
1. **get access**: add (subj, obj, access-mode) to b
   - used by a subj to initiate an access to an object
2. **release access**: remove (subj, obj, access-mode)
3. **change object level**
4. **change current** level
5. **give access permission**: Add an access mode to M
   - used by a subj to grant access to on an obj
6. **rescind access permission**: reverse of 5
7. **create an object**
8. **delete a group of objects**

BLP Example

- A role-based access control system
- Two users: Carla (student) and Dirk (teacher)
  - Carla (Class: s)
  - Dirk (Class: T); can also login as a students thus (Class: s)
- A student role has a lower security clearance
- A teacher role has a higher security clearance

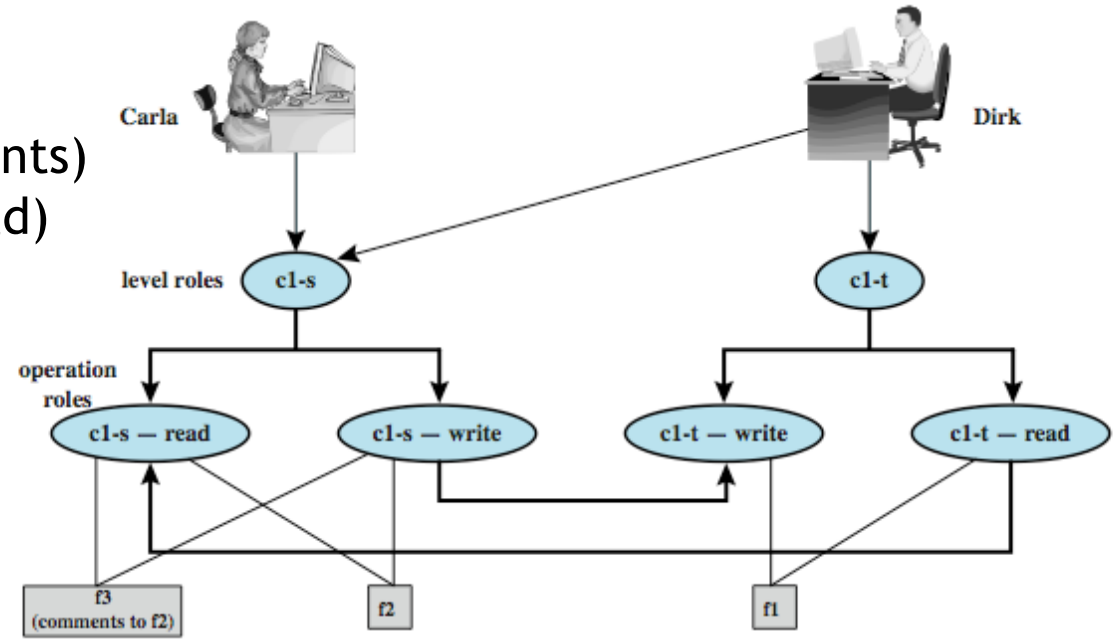
BLP Example

Dirk creates f1; Carla creates f2
Carla can read/write to f2
Carla can't read f1
Dirk can read/write f1
Dirk can read f2 (if perm)
Dirk can read/write f2 only as a stu



(a) Two new files are created: f1: c1-t; f2: c1-s

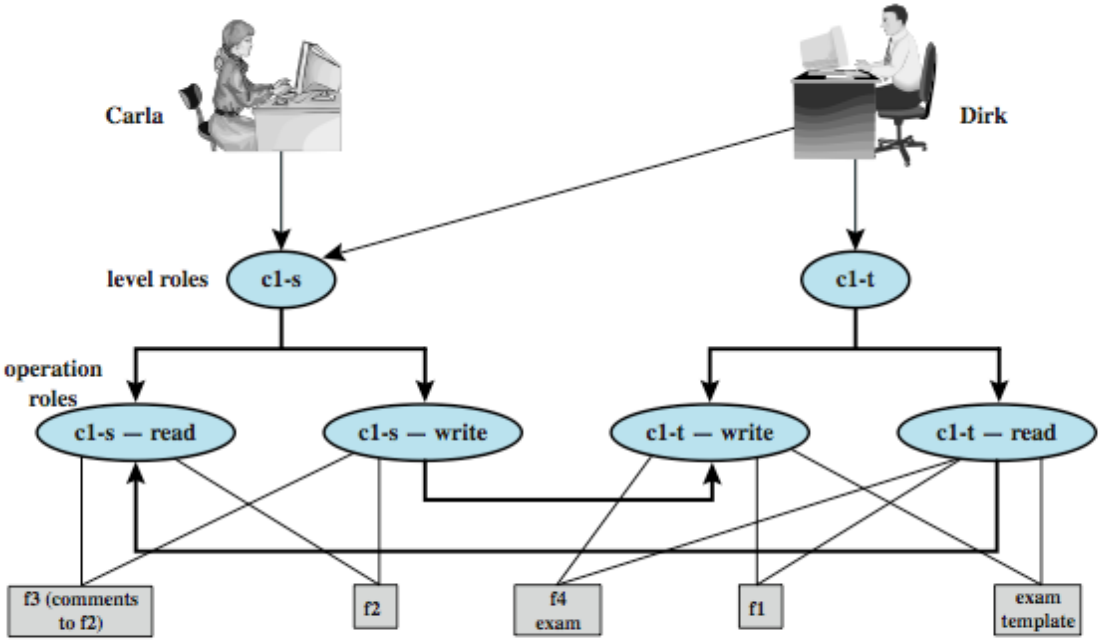
Dirk reads f2; want create f3 (comments)
Dirk signs in as a stu (so Carla can read)
As a teacher, Dirk cannot create a file at stu classification



(b) A third file is added: f3: c1-s

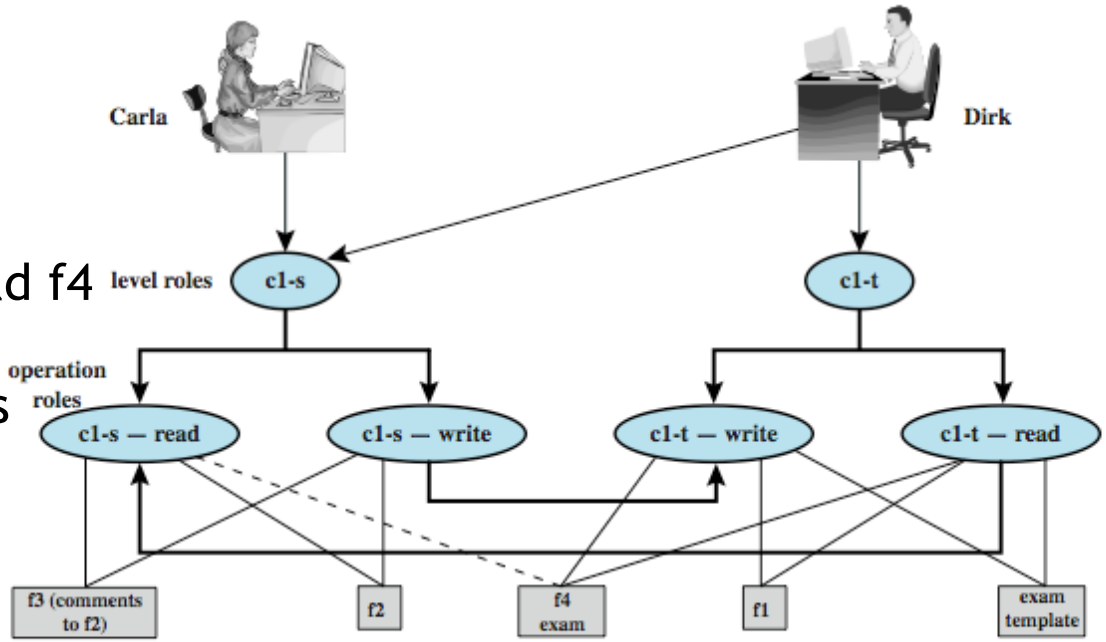
BLP Example cont.

Dirk as a teacher creates exam (f4)
Must log in as a teacher to read
template



(c) An exam is created based on an existing template: f4: c1-t

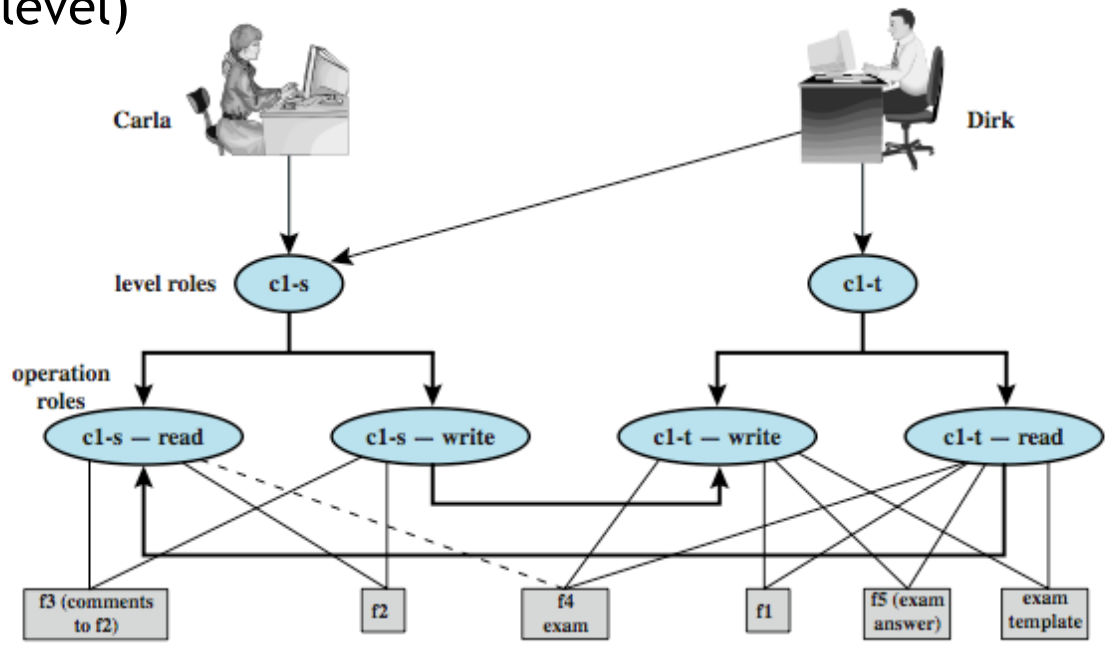
Dirk wants to give Carla access to read f4
Dirk can't do that; an admin must do
An admin downgrades f4 class to c1-s



(d) Carla, as student, is permitted acess to the exam: f4: c1-s

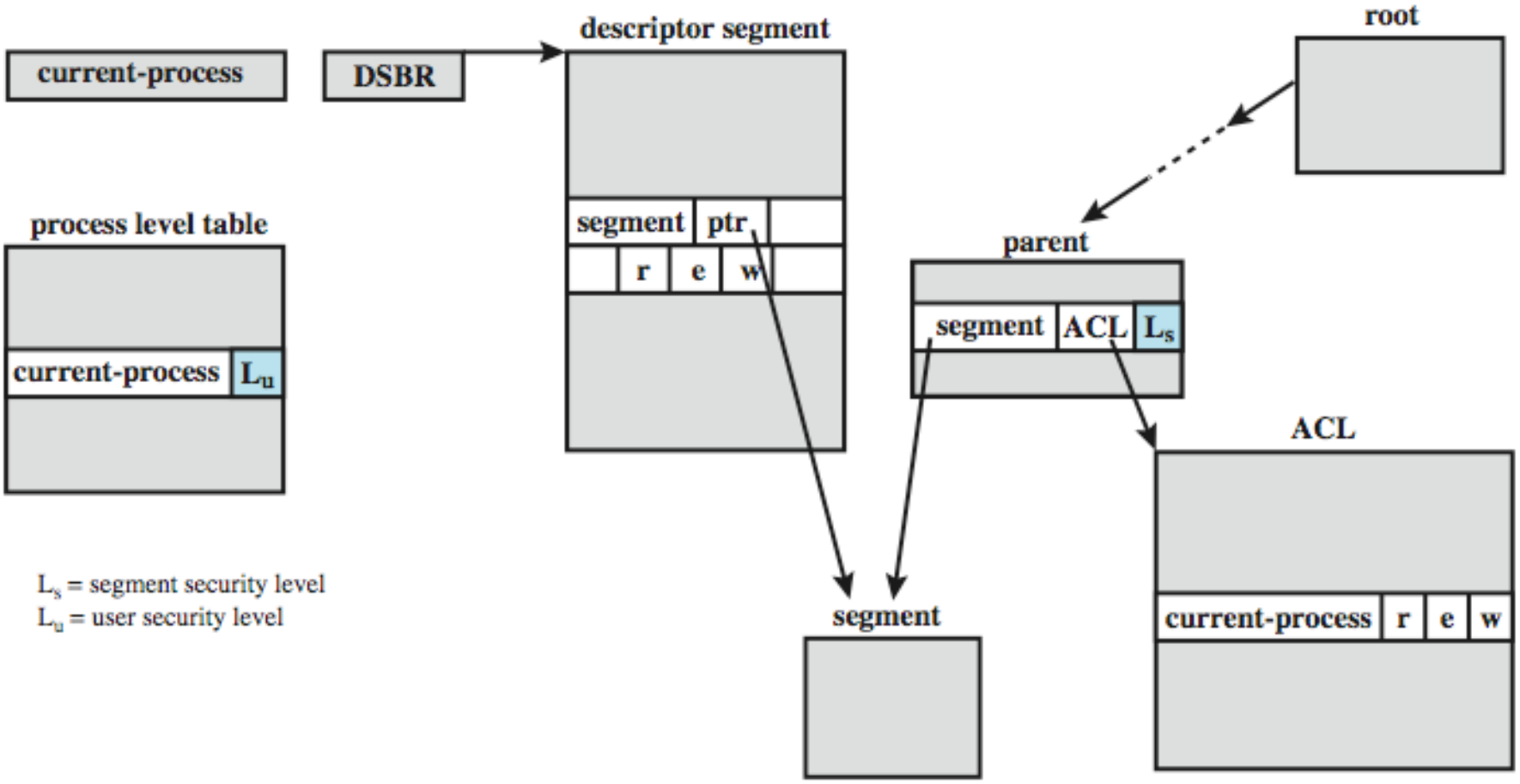
BLP Example cont.

Carla writes answers to f5 (at c1-t level)
 -- An example of write up
Dirk can read f5



(e) The answers given by Carla are only accessible for the teacher: f5: c1-t

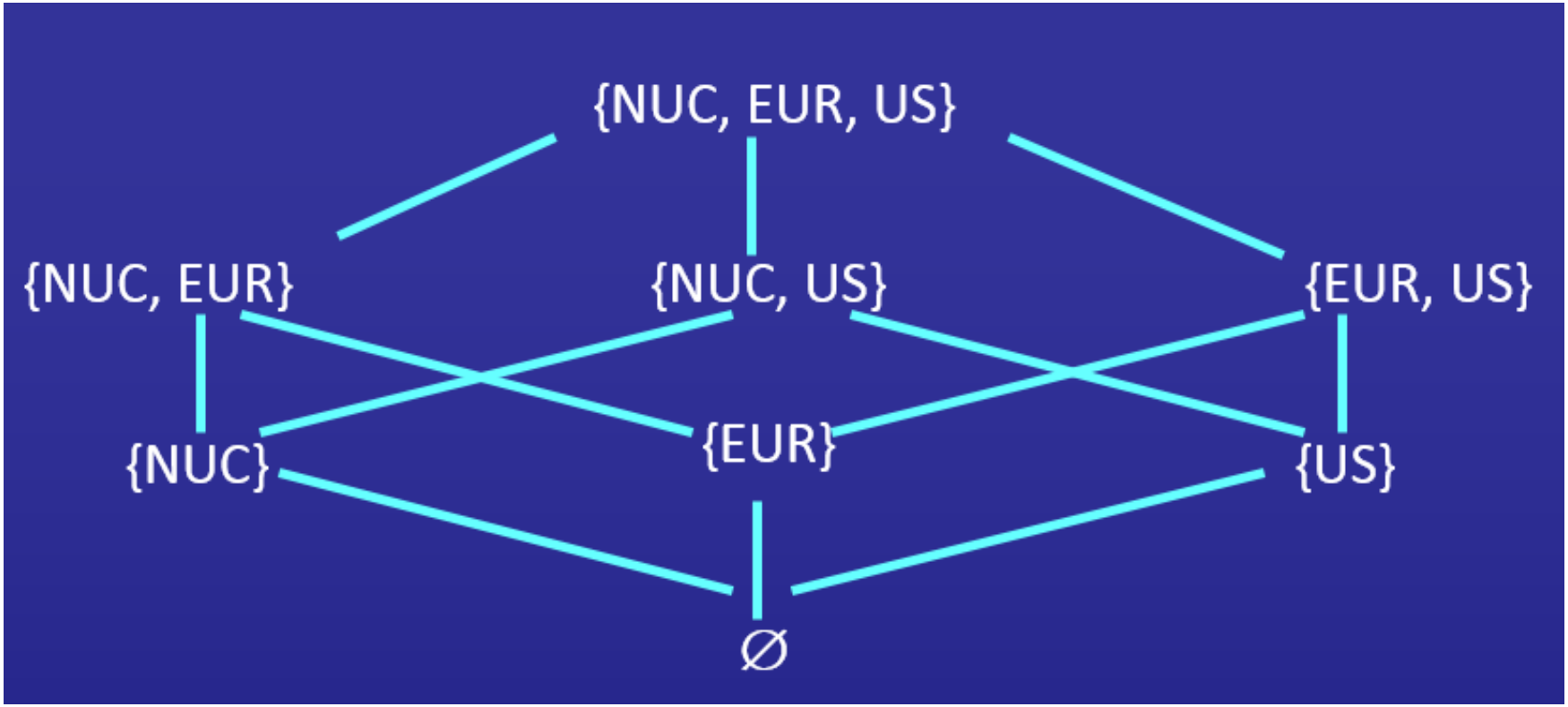
MULTICS Example



BLP Categories

- Expand the model to add categories to each security classification
- Objects placed in multiple categories
- Based on “need to know” principle
- Example categories: NUC, EUR, US
  - One can have access to any of these: none, {NUC}, {EUR}, {US}, {NUC, EUR}, … {NUC, EUR, US}
  - Categories form a lattice under the “subset of” operation

The Lattice Hierarchy



BLP Dominate (dom) Relationship

- Captures the combination of security classification and category set
- (A, C) *dom* (A', C') iff $A' \leq A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) $\neg$*dom* (Confidential, {EUR})

An Example of dom Relationship

- George is cleared into security level (S, {NUC, EUR})
- `DocA` is classified as (C, {NUC})
- `DocB` is classified as (S, {EUR, US})
- `DocC` is classified as (S, {EUR})
- George dom `DocA`
- George ¬dom `DocB`
- George dom `DocC`

Reading Information - New

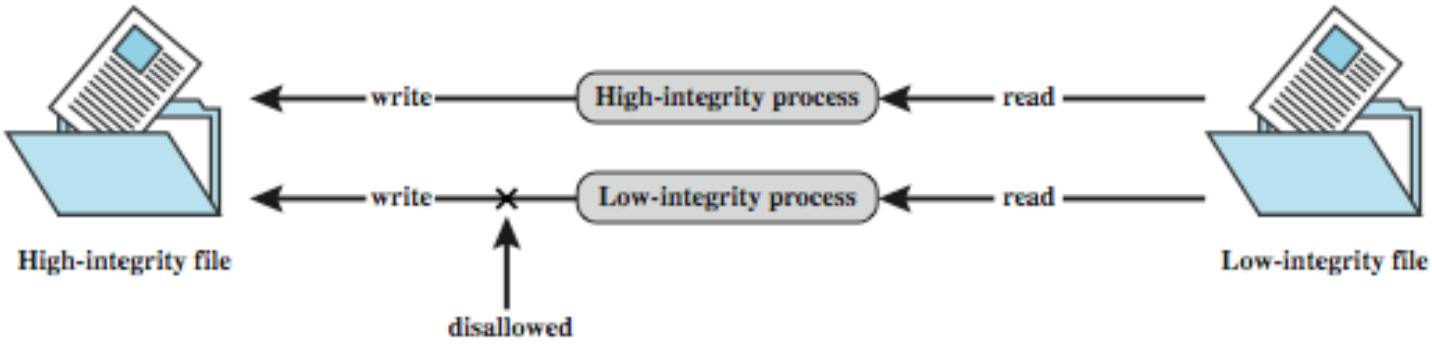
- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition
  - Subject *s* can read object *o* iff *L*(*s*) *dom L*(*o*) and *s* has permission to read *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information - New

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 2)
  - Subject *s* can write object *o* iff *L*(*o*) *dom L*(*s*) and *s* has permission to write *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

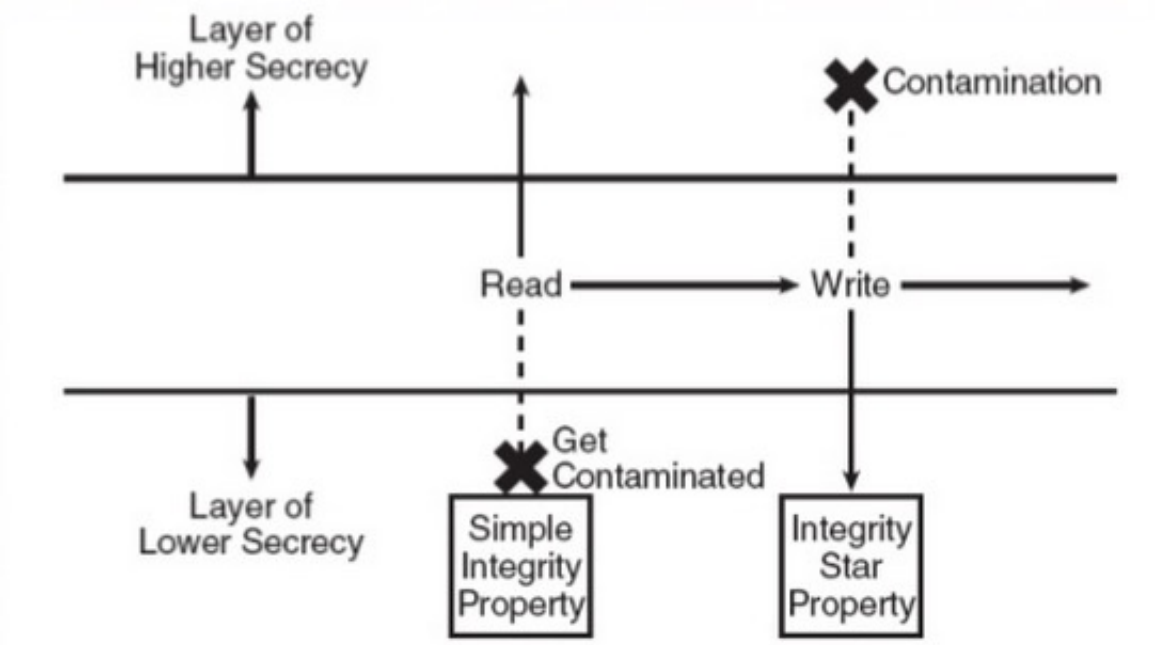
Biba Integrity Model

- Various models dealing with integrity
- Strict integrity policy:
  - Simple integrity: *modify only if* $I(S) \geq I(O)$
  - Integrity confinement: *read only if* $I(S) \leq I(O)$
  - Invocation property: *invoke/comm only if* $I(S_1) \geq I(S_2)$



Biba Integrity Model

- Simple integrity: *modify only if* $I(S) \geq I(O)$
- Integrity confinement: *read only if* $I(S) \leq I(O)$
- Invocation property: *invoke/comm only if* $I(S_1) \geq I(S_2)$



Clark-Wilson Integrity Model

- Two concepts
  - Well-formed transactions: a user can manipulate data in constrained ways
  - Separation of duty: one can create a transaction but not execute it
- CDI: constrained data items (loan app; checks)
- UDI: unconstrained items
- IVPs: procedures that assure all CDIs conform to integrity/consistency rules
- TPs: transactions that change CDIs
- Very practical; used in commercial world

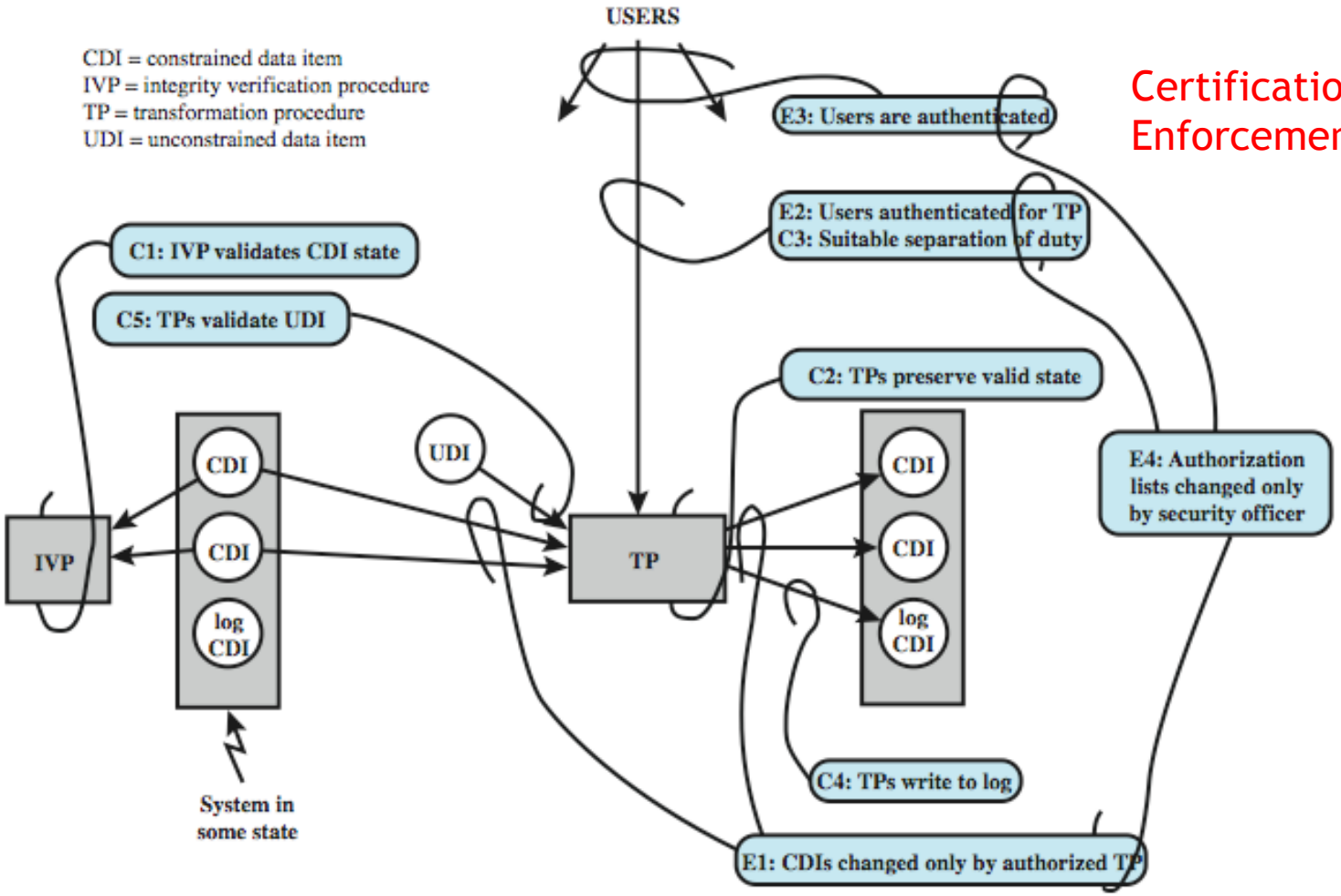
Certified and Enforcement Rules

- C1: IVPs must ensure that all CDIs are in valid states
- C2: All TPs must be certified (must take a CDI from a valid state to a valid final state)
  - (Tpi, CDIa, CDIb, CDIc, …)
- E1: The system must maintain a list of relations specified in C2
- E2: The system must maintain a list of (User, Tpi, (CDIa, CDIb, …))

Certified and Enforcement Rules

- C3: The list of relations in E2 must be certified to meet separation of duties
- E3 The system must authenticate each user when executing a TP
- C4: All TPs must be certified
- C5: Any TP that takes UDI as in input value must be certified to perform valid transaction
- E4: Only the agent permitted to certify entitles is allowed to do so

Clark-Wilson Integrity Model



Certification (C) and Enforcement (E) rules

The Chinese Wall Model

- Also known as Brewer and Nash model
- Hybrid model: addresses integrity and confidentiality
- Addresses conflict of interest (CI or CoI)
- Model elements
  - **subjects**: active entities interested in accessing protected objects
  - **information**
    - objects: individual data items, each about a corp
    - datasets (DS): all objects concerning one corp
    - CI class: datasets whose corp are in competition (conflict of interest or CI)
  - **access rules**: rules for reading/writing data

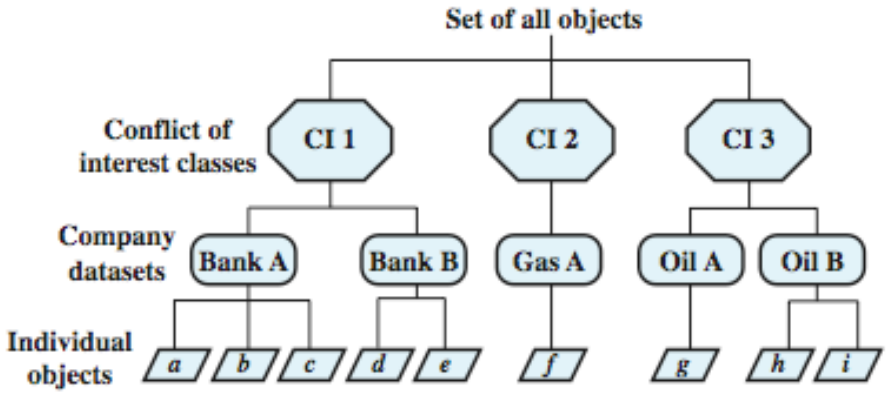
The Chinese Wall Model

- Not a true multilevel secure model
  - the history of a subject's access determines access control
- Subjects are only allowed access to info that is not held to conflict with any other info they already possess
- Once a subject accesses info from one dataset, a *wall* is set up to protect info in other datasets in the same CI

Chinese Wall Model

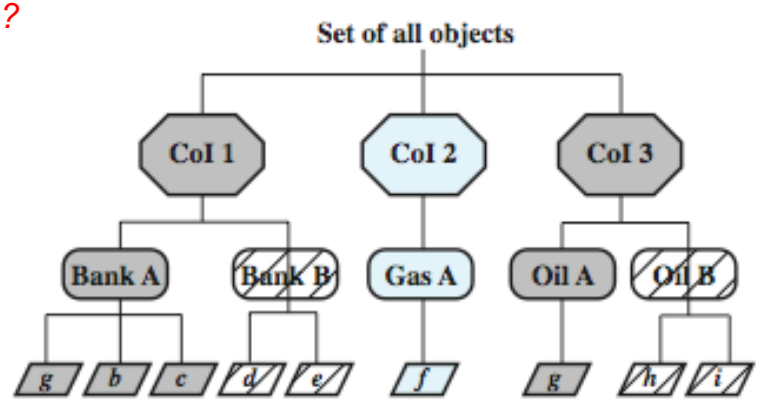
Simple sec rule (read): S can read O if O is in the same DS as an object already accessed by S OR O belongs to a CoI from which S has not yet accessed any info

***-property (write):** S can write O only if S can read O and all objects that S can read are in the same DS as O.

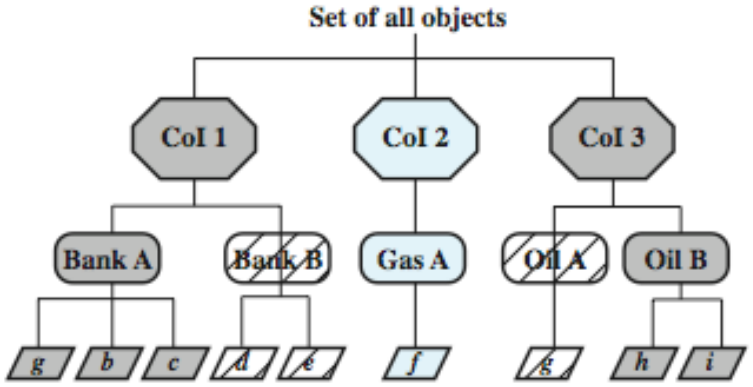


(a) Example set

Question: what can John or Jane write to?



(b) John has access to Bank A and Oil A

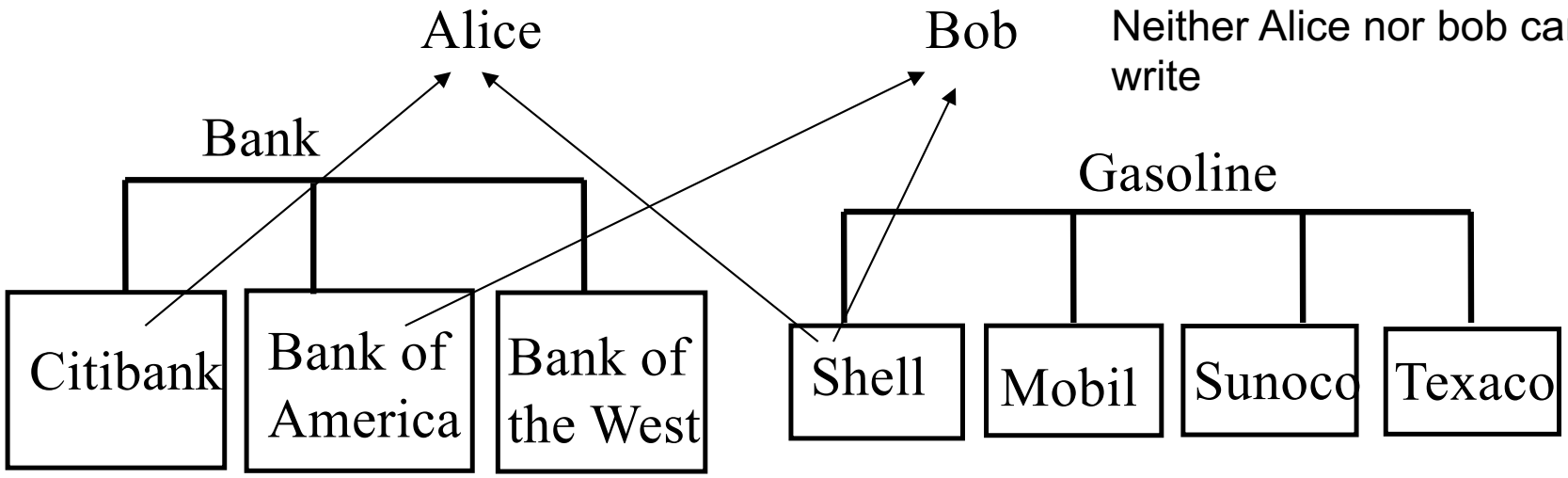


(c) Jane has access to Bank A and Oil B

CW-*-Property

s can write to *o* iff both of the following hold:

1. The CW-simple condition permits *s* to read *o*
2. For all *unsanitized* objects *o′*, if *s* can read *o′*, then *CD* (*o′*) = *CD* (*o*)
   - All *s* can read are either within the same CD, or sanitized

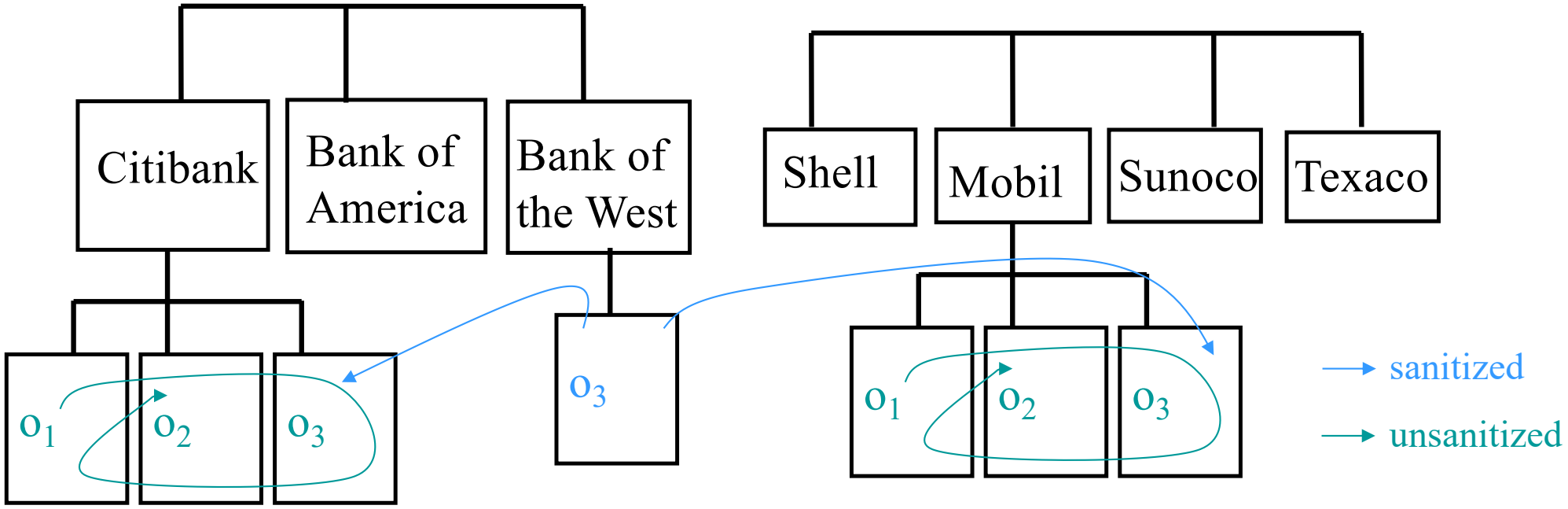


How Does Information Flow?

- With the two conditions (CW simple security condition and CW *-property) in place, how can information flow around the system?
- Main Results
  - In each COI class (e.g. Bank), a subject can only read objects in a single CD (e.g. Citibank)
  - At least *n* subjects are required to access all objects in a COI class with totally *n* CDs

How Does Information Flow? (Cont'd)

- Information flows from *o* to *o'* if *s* reads *o* and writes *o'*
- information in an unsanitized object can only flow inside that CD; information in sanitized objects can flow freely



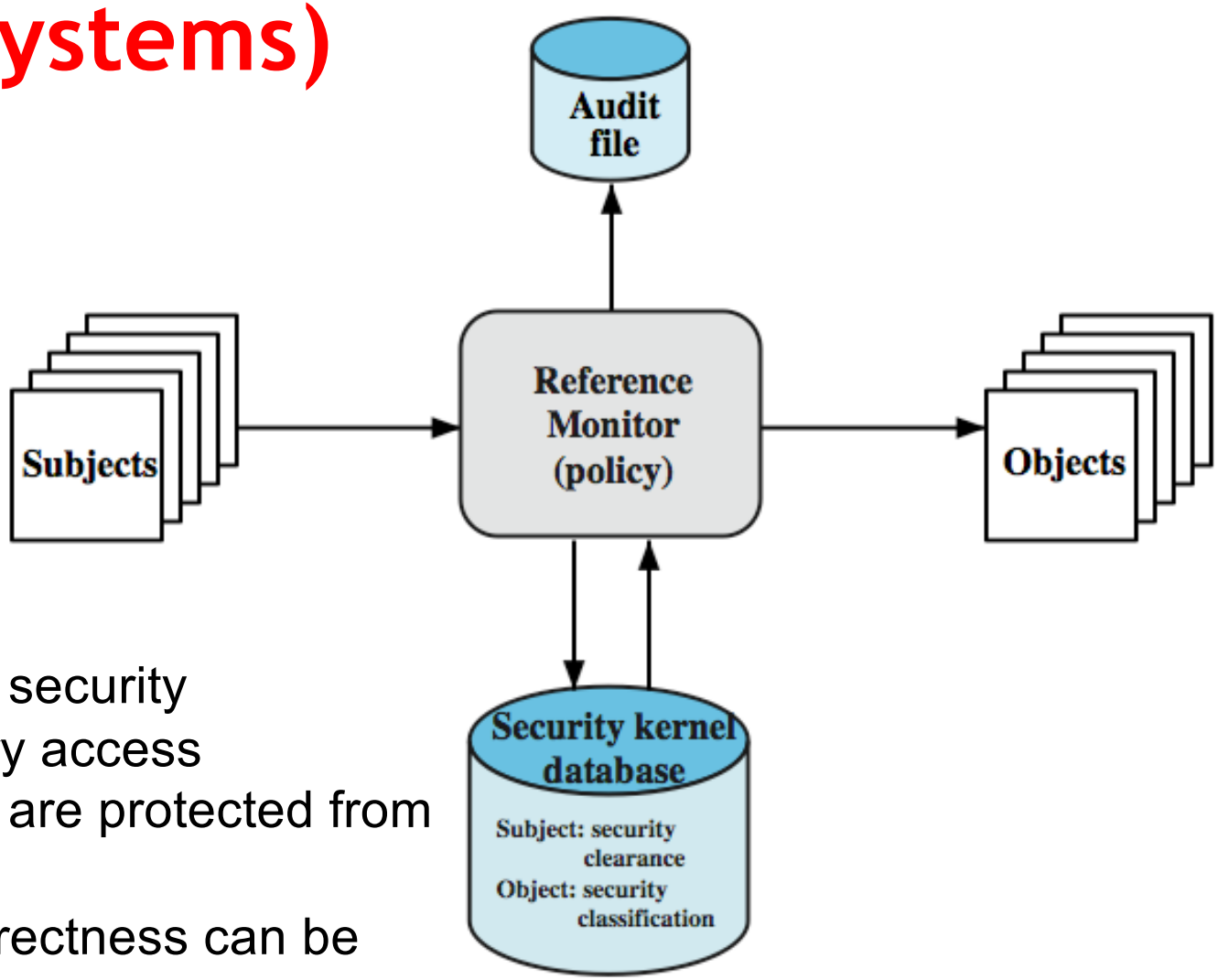
Compare CW to Bell-LaPadula

- CW is based on access history, BLP is history-less
- BLP can capture CW state at any time, but cannot track changes over time
  - BLP security levels would need to be updated each time an access is allowed

Trusted Systems

- **Trusted system**: A system believed to enforce a given set of attributes to a stated degree of assurance
- **Trustworthiness**: Assurance that a system deserves to be trusted, such that the trust can be guaranteed in some convincing way, such as through formal analysis or code review
- **Trusted computer system**: A system that employs sufficient hardware and software assurance measures to allow its use for simultaneous processing of a range of sensitive or classified information

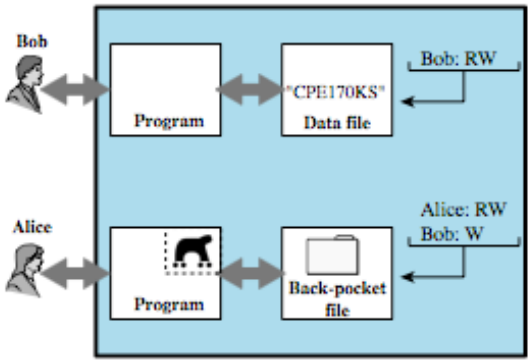
Reference Monitors (Trusted Systems)

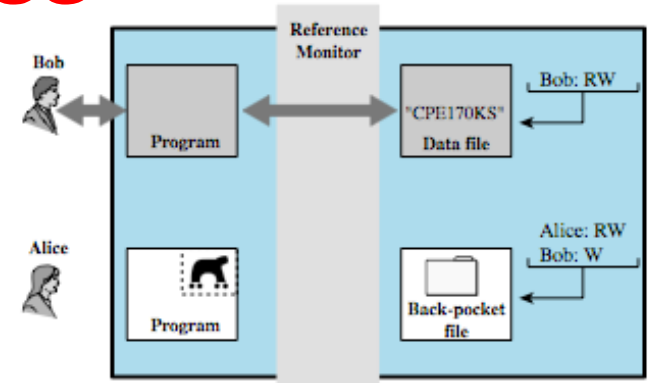


Properties of the RM:

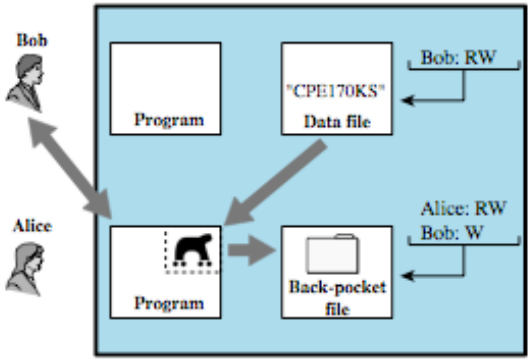
Complete mediation: security rules enforced on every access
Isolation: RM and DB are protected from unauthorized access
Verifiability: RM's correctness can be proven

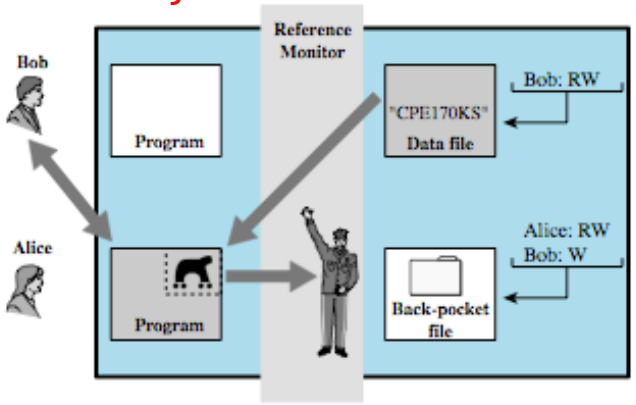
Trojan Horse Defence





security levels are assigned at login
sec levels: sensitive and public data
Trojan must have the same sec level





A “normal” OS vs a trusted OS with RM
(sec levels assigned at login thus can’t write down)

MLS Security for Role-Based Access Control

- Role-based access control (RBAC) can implement BLP MLS rules given:
  - Security constraints on users:

    For all users u, sec-level(u) is defined
  - Constraints on read/write permissions:

    All objects have a defined r and w access permission
  - Read and write level role access definitions

    Each role r defined has r-level(r) and w-level(r)
  - Constraint on user-role assignments

    Clearance of a user must dominate the r-level and be dominated by the w-level

MLS in Database Security

- Classification by table and column

Department Table - U		
Did	Name	Mgr
4	accts	Cathy
8	PR	James

Employee - R			
Name	Did	Salary	Eid
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

(a) Classified by table

Department Table		
Did -U	Name -U	Mgr -R
4	accts	Cathy
8	PR	James

Employee			
Name -U	Did -U	Salary -R	Eid -U
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

MLS in Database Security

- Classification by row and cell

Department Table			
Did	Name	Mgr	
4	accts	Cathy	R
8	PR	James	U

Employee				
Name	Did	Salary	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
Ziggy	8	67K	3054	R

(c) Classified by row (tuple)

Department Table		
Did	Name	Mgr
4 - U	accts - U	Cathy - R
8 - U	PR - U	James -R

Employee			
Name	Did	Salary	Eid
Andy - U	4 - U	43K - U	2345 - U
Calvin - U	4 - U	35K - U	5088 - U
Cathy - U	4 - U	48K - U	7712 - U
James - U	8 - U	55K - R	9664 - U
Ziggy - U	8 - U	67K - R	3054 - U

Database Security: Read Access

- DBMS enforces simple security rule (no read up)
- Easy if granularity is entire database or at table level
- Inference problems if have column granularity
  - If can query on restricted data can infer its existence
  - `SELECT Ename FROM Employee`
  - `SELECT Ename FROM Employee WHERE Salary > 50K`
  - Solution is to check access to all query data
- Also have problems if have row granularity
  - Null response indicates restricted/empty result
- No extra concerns if have element granularity

Database Security: Write Access

- Enforce *-security rule (no write down)
- Have problem if a low clearance user wants to insert a row with a primary key that already exists in a higher level row:
  - Can reject, but user knows row exists
  - Can replace, compromises data integrity
  - Polyinstantiation and insert multiple rows with same key, creates conflicting entries but plausible (and implemented in many DBs)
- Same alternatives occur on update
- Avoid problem if use database/table granularity

Trusted Computing

Trusted Computing encompasses six key technology concepts, of which all are required for a fully Trusted system:

1. Endorsement key
2. Secure input and output
3. Memory curtaining / protected execution
4. Sealed storage
5. Remote attestation
6. Trusted Third Party (TTP)

Trusted Computing

1. Endorsement key
2. Secure input and output
3. Memory curtaining / protected execution
4. Sealed storage
5. Remote attestation
6. Trusted Third Party (TTP)

Endorsement Key

- The endorsement key is a 2048-bit RSA public and private key pair that is created randomly on the chip at manufacture time and cannot be changed.

- The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip

- This key is used to allow the execution of secure transactions: every Trusted Platform Module (TPM) is required to be able to sign a random number (in order to allow the owner to show that he has a genuine trusted computer),

Physically Unclonable Function (PUF)

- A PUF is a physical object whose operation cannot be reproduced ("cloned") in physical way, that for a given input and conditions (challenge), provides a physically defined "digital fingerprint" output (response). that serves as a unique identifier
- PUFs are often based on unique physical variations occurring naturally during semiconductor manufacturing
- PUFs depend on the uniqueness of their physical microstructure. This microstructure depends on random physical factors introduced during manufacturing. These factors are unpredictable and uncontrollable, which makes it virtually impossible to duplicate or clone the structure.

Physically Unclonable Function (PUF)

- Rather than embodying a single cryptographic key, PUFs implement challenge-response-authentication to evaluate this microstructure.
- When a physical stimulus is applied to the structure, it reacts in an unpredictable (but repeatable) way due to the complex interaction of the stimulus with the physical microstructure of the device.
- The applied stimulus is called the challenge, and the reaction of the PUF is called the response.
- A specific challenge and its corresponding response together form a challenge-response pair or CRP. The device's identity is established by the properties of the microstructure itself. As this structure is not directly revealed by the challenge-response mechanism, such a device is resistant to spoofing attacks.

Memory Curtaining

- Memory curtaining extends common memory protection techniques to provide full isolation of sensitive areas of memory—for example, locations containing cryptographic keys.
- Even the operating system does not have full access to curtained memory

Sealed Storage

- Sealed storage protects private information by binding it to platform configuration information including the software and hardware being used.
- This means the data can be released only to a particular combination of software and hardware.

Remote Attestation

- Remote attestation allows changes to the user's computer to be detected by authorized parties.
- For example, software companies can identify unauthorized changes to software, including users modifying their software to circumvent commercial digital rights restrictions.
- The hardware generates a certificate stating what software is currently running. The computer can then present this certificate to a remote party to show that unaltered software is currently executing

Trusted Third Party

A trusted third party (TTP) is an entity which facilitates interactions between two parties who both trust the third party; the third party reviews all critical transaction communications between the parties, based on the ease of creating fraudulent digital content.

In TTP models, the relying parties use this trust to secure their own interactions

Trusted Platform Module (TPM)

- Concept from Trusted Computing Group
- Hardware module at heart of hardware/ software approach to trusted computing
- Uses a TPM chip on
  - motherboard, smart card, processor
  - working with approved hardware / software
  - generating and using crypto keys
- Has 3 basic services: authenticated boot, certification, and encryption

Trusted Platform Module (TPM)

- Concept from Trusted Computing Group
- A Trusted Platform Module (TPM) is a secure cryptoprocessor that implements the ISO/IEC 11889 standard.
- Common uses are:
  - Verifying that the boot process starts from a trusted combination of hardware and software
  - Storing encryption keys

TPM Functions

- A hardware random number generator
- Facilities for the secure generation of cryptographic keys for limited uses.
- Remote attestation: Creates a nearly unforgeable hash key summary of the hardware and software configuration.
  - One can use the hash to verify that the hardware and software have not been changed. The software in charge of hashing the setup determines the extent of the summary.

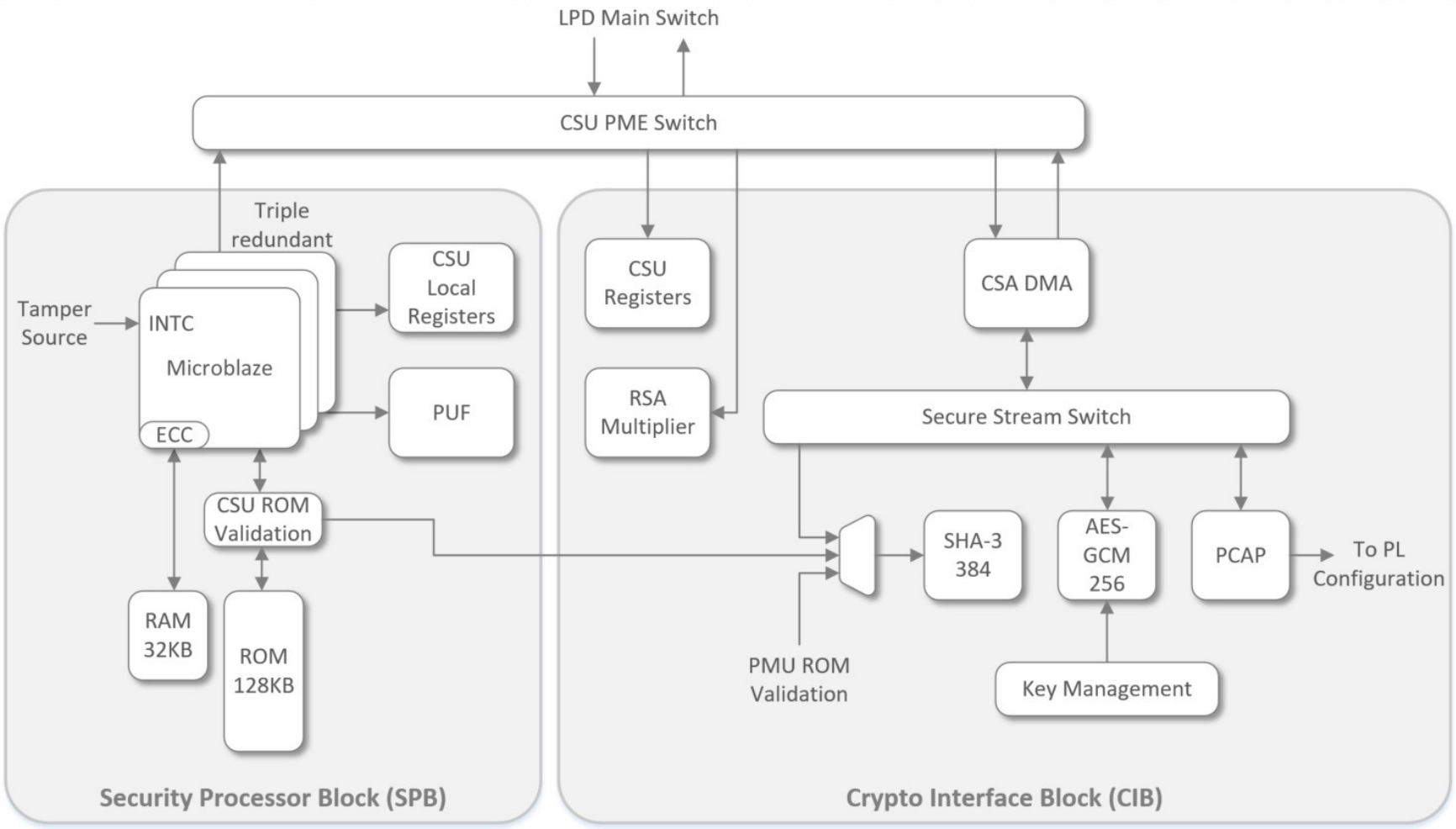
TPM Functions

Binding: Data is encrypted using the TPM bind key, a unique RSA key descended from a storage key. Computers that incorporate a TPM can create cryptographic keys and encrypt them so that they can only be decrypted by the TPM. This process, often called wrapping or binding a key, can help protect the key from disclosure. Each TPM has a master wrapping key, called the storage root key, which is stored within the TPM itself. User-level RSA key containers are stored with the Windows user profile for a particular user and can be used to encrypt and decrypt information for applications that run under that specific user identity.

Sealed storage: Specifies the TPM state for the data to be decrypted (unsealed).

Other Trusted Computing functions for the data to be decrypted (unsealed).

Xilinx Ultrascale+ MPSoC Security Unit Architecture

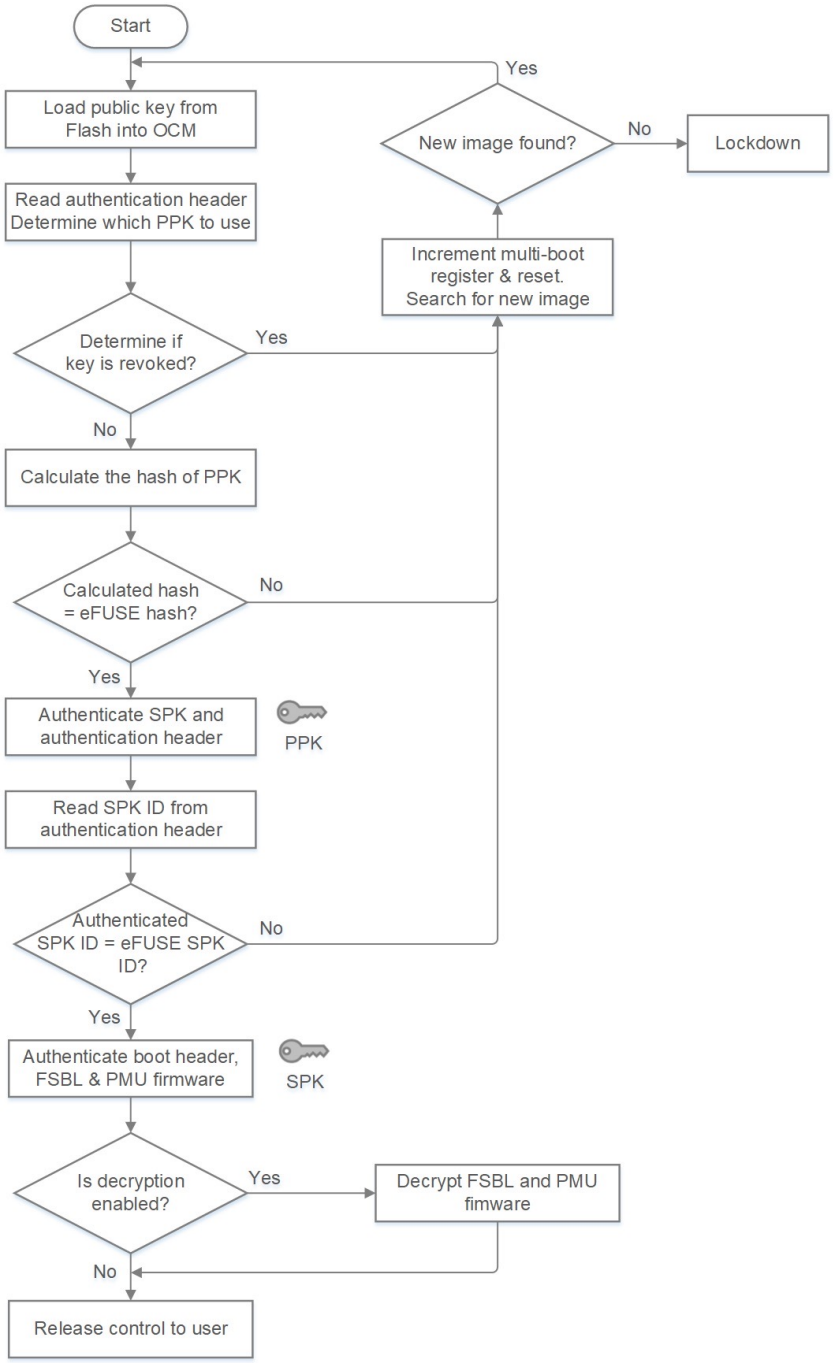


Secure Boot

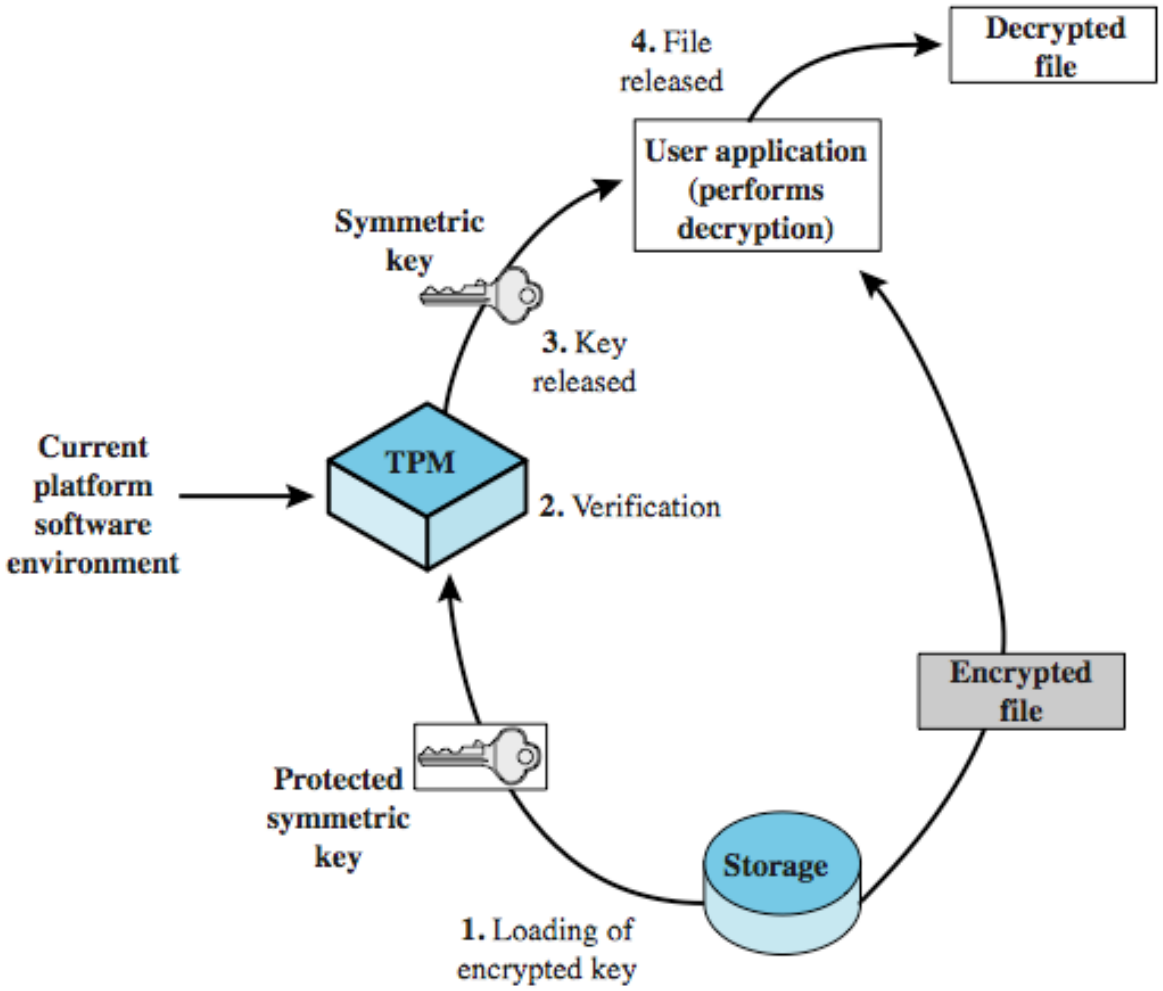
The secure boot is accomplished by using the hardware root of trust boot mechanism, which also provides a way to encrypt all of the boot or configuration files.

This architecture provides the required confidentiality, integrity, and authentication to host the most secure of applications.

The Zynq UltraScale+ MPSoC hardware root of trust is based on the RSA-4096 asymmetric authentication algorithm in conjunction with SHA-3/384. There are two key pairs used in the Zynq UltraScale+ MPSoC, and consequently two public key types: the primary public key (PPK) and the secondary public key (SPK).



Protected Storage Function



Trusted Systems

- security models aimed at enhancing trust
- work started in early 1970's leading to:
  - Trusted Computer System Evaluation Criteria (TCSEC), Orange Book, in early 1980s
  - further work by other countries
  - resulting in Common Criteria in late 1990s
- also Computer Security Center in NSA
  - with Commercial Product Evaluation Program
  - evaluates commercially available products
  - required for Defense use, freely published

Common Criteria (CC)

- ISO standards for security requirements and defining evaluation criteria to give:
  - Greater confidence in IT product security
  - Formal actions during process of:
    - development using secure requirements
    - evaluation confirming meets requirements
    - operation in accordance with requirements
- Evaluated products are listed for use

CC Requirements

- Have a common set of potential security requirements for use in evaluation
- Target of evaluation (TOE) refers product/system subject to evaluation
- Functional requirements
  - define desired security behavior
- Assurance requirements
  - that security measures effective correct
- Requirements: see pages 471-472

Assurance

- “Degree of confidence that the security controls operate correctly and protect the system as intended”
- Applies to:
  - product security requirements, security policy, product design, implementation, operation
- various approaches analyzing, checking, testing various aspects

Common Criteria (CC) Assurance Levels

- EAL 1: functionally *independently* tested
- EAL 2: structurally tested *(includes review of design and vulnerability analysis)*
- EAL 3: methodically tested and checked *(design testing)*
- EAL 4: methodically designed, tested, and reviewed *(high level to low level vulnerability analysis)*
- EAL 5: semiformally designed and tested
- EAL 6: semiformally verified design and tested
- EAL 7: formally verified design and tested (formal analysis and formally showing correspondence)

Evaluation Parties & Phases

- Evaluation parties:
  - sponsor - customer or vendor
  - developer - provides evidence for evaluation
  - evaluator - confirms requirements satisfied
  - certifier - agency monitoring evaluation process
- Phases:
  - preparation (initial contact)
  - conduct of evaluation (structured process)
  - conclusion (final evaluation)
- Government agency regulates: NIST, NSA jointly operate Common Criteria Eval and Validation Scheme (US CCEVS)

Cryptographic Module Validation Program (CVMP)

- The National Institute of Standards and Technology (NIST) issued the FIPS 140 Publication Series to coordinate the requirements and standards for cryptography modules that include both hardware and software components.
- The Cryptographic Module Validation Program (CMVP) is a joint effort between the National Institute of Standards and Technology under the Department of Commerce and the Canadian Centre for Cyber Security
- CMVP validates cryptographic modules to Federal Information Processing Standard (FIPS) 140-3

Summary

- Bell-LaPadula security model
- other models
- Reference monitors & trojan horse defence
- multilevel secure RBAC and databases
- Trusted platform module
- Common criteria
- Assurance and evaluation