COMP 3355
Access Control

K.P. Chow
University of Hong Kong

Objectives

- Windows access control
- Access control models
- Accountability principles

Access Control

- Deals with the prevention and detection of unauthorized actions by users of a computer system
- 2 main issues:
  - *Identification and* ***authentication***: stating who you are and proving the identification claim with credentials
  - ***Authorization*** *and access control*: restriction on the ability of a subject to use a system or an object in that system

Windows Authentication

- Username and password are used for authentication
- Windows 2000 uses 2 authentication packages: MSV1_0 and Kerberos
- **MSV1_0**:
  - Default authentication package on a stand-alone Windows 2000 system
  - LSA uses MSV1_0 on domain-member computers to authenticate pre-Windows 2000 domains and computers that cannot locate a domain controller
- **Kerberos**:
  - Used on computers that are members of Windows 2000 domains

Windows MSV1_0 Authenication

- MSV1_0 takes the username and the hashed password and sends a request to the Security Account Manager (SAM) module to retrieve the account information
- MSV1_0 then compares the hashed password and the username to that of stored in the SAM, if matches, a logon session is created
- The encrypted passwords are stored in the user accounts which are held in the SAM database:
  - The SAM database is part of the **registry**
  - A binary file, accessed using SAM APIs
  - Each password is hashed using a one-way function, i.e. password cannot be retrieved in plain form

CTRL+ALT+DEL

- Always press CTRL+ALT+DEL when starting a Windows session, WHY?

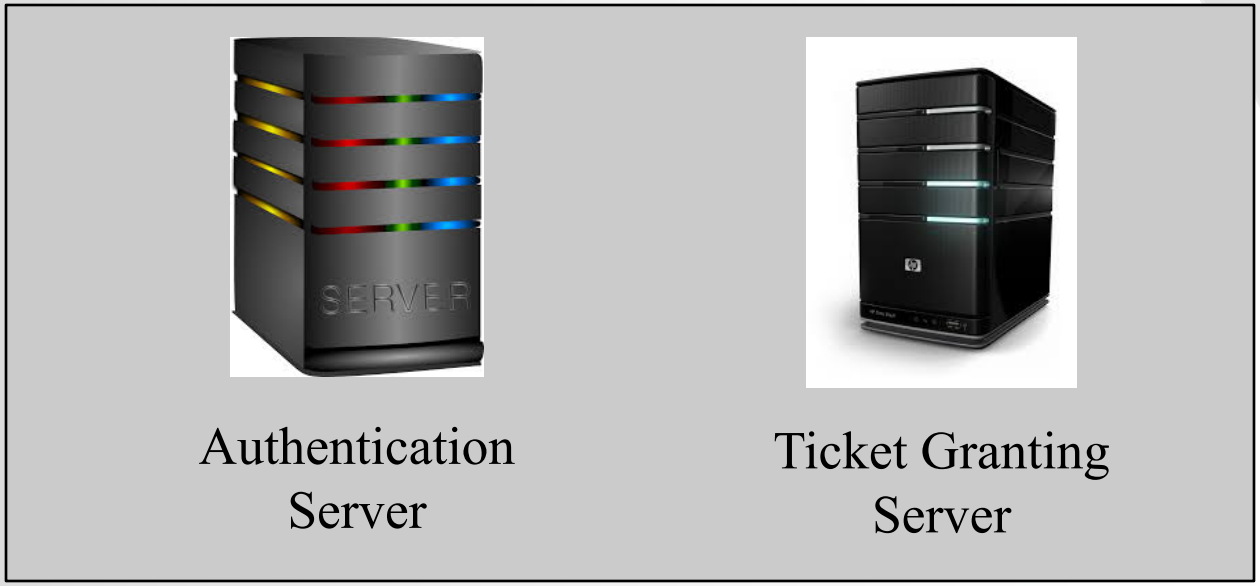
CTRL+ALT+DEL

- Aka “secure attention key”: generates calls to low-level functions that cannot be duplicated by application programs
- Invokes the Windows operating system logon screen
- Provides a *trusted path* from the keyboard to the login process (winlogon.exe)

Kerberos

- Key Distribution Center (KDC):
  - Authenticates users at login and issues tickets which are valid for one session
  - Enables users to obtain other tickets from ticket-granting servers
  - Maintains a database of secret keys of all users
  - 2 components: authentication server (AS) and ticket granting server (TGS)

KDC

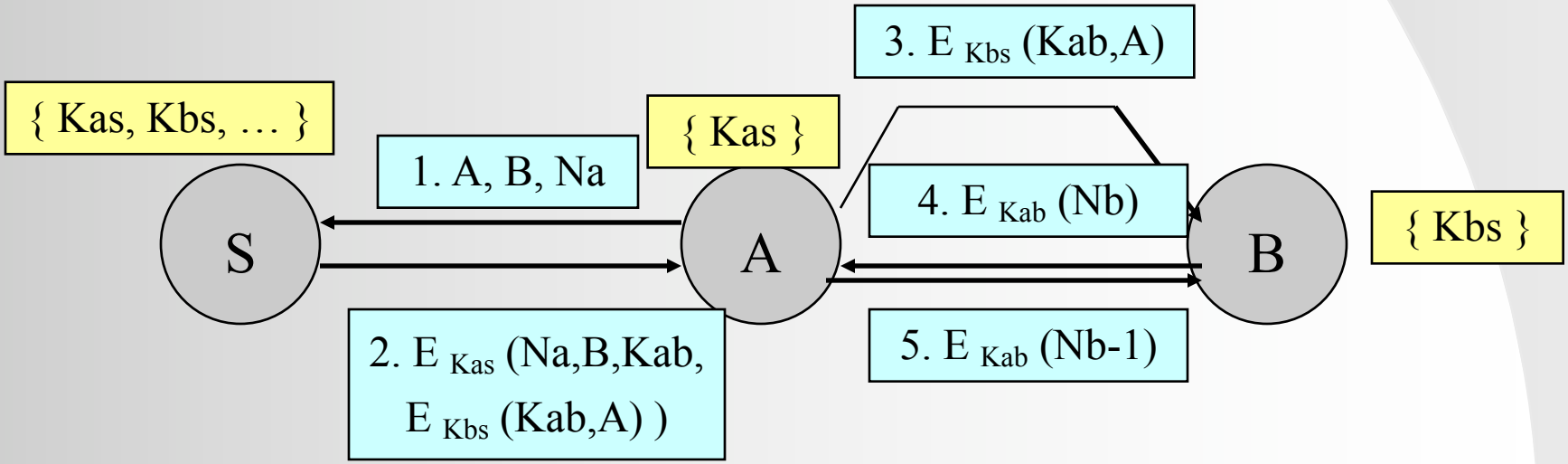


KDC

- Authentication server (AS): also called Kerberos Server
  - A user presents an authenticating credential (e.g. password) to the authentication server
  - AS returns a ticket to the user showing that the user has passed authentication
- Ticket Granting Server (TGS):
  - A users wants to access a resource R (e.g. a file), he sends his authenticated ticket and a request to use R to TGS
  - The TGS returns 2 tickets to the user: one shows that the user's access to R is authorized, the other is for the user to present to R in order to access R
- Can be used to implement single sign-on

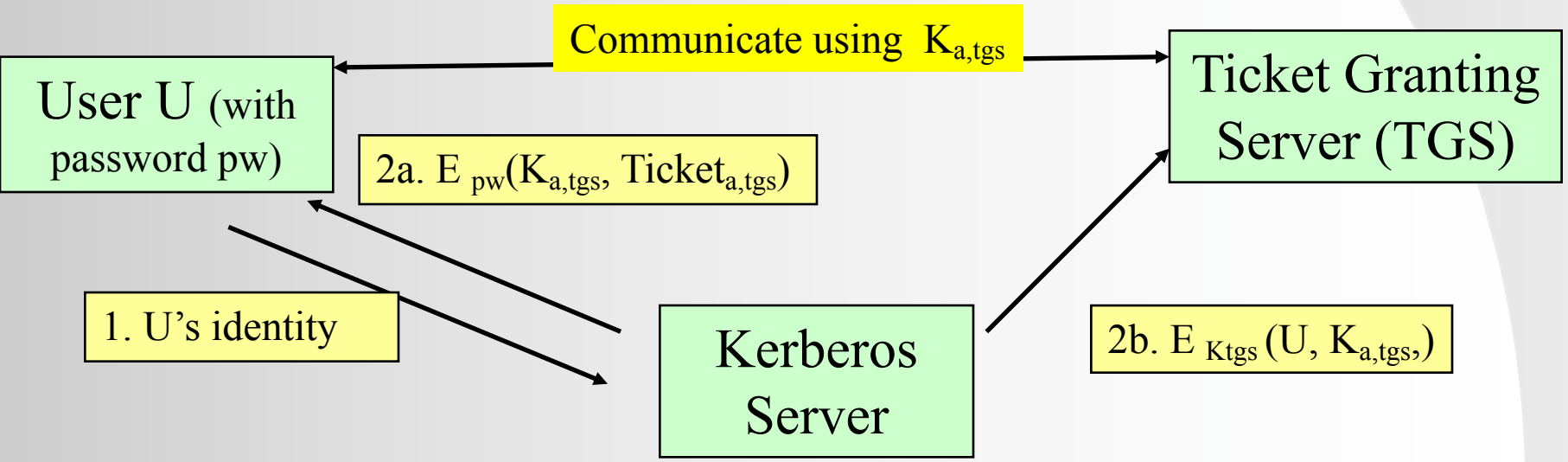
Needham and Schroeder Protocol

- A key exchange protocol used in Kerberos
- Based on symmetric key crytography
- Let
  - Kab: session key between A and B
  - Na, Nb: random numbers
  - E_K (M): encrypt M using K



Kerberos Authentication Protocol
Initiating a Kerberos Session

- User U with password pw
- Kerberos server: store (U, pw)
- K_{Ktgs} : the secret key shared between Kerbers server and TGS
- $K_{a,tgs}$: the session key to be used between U and TGS

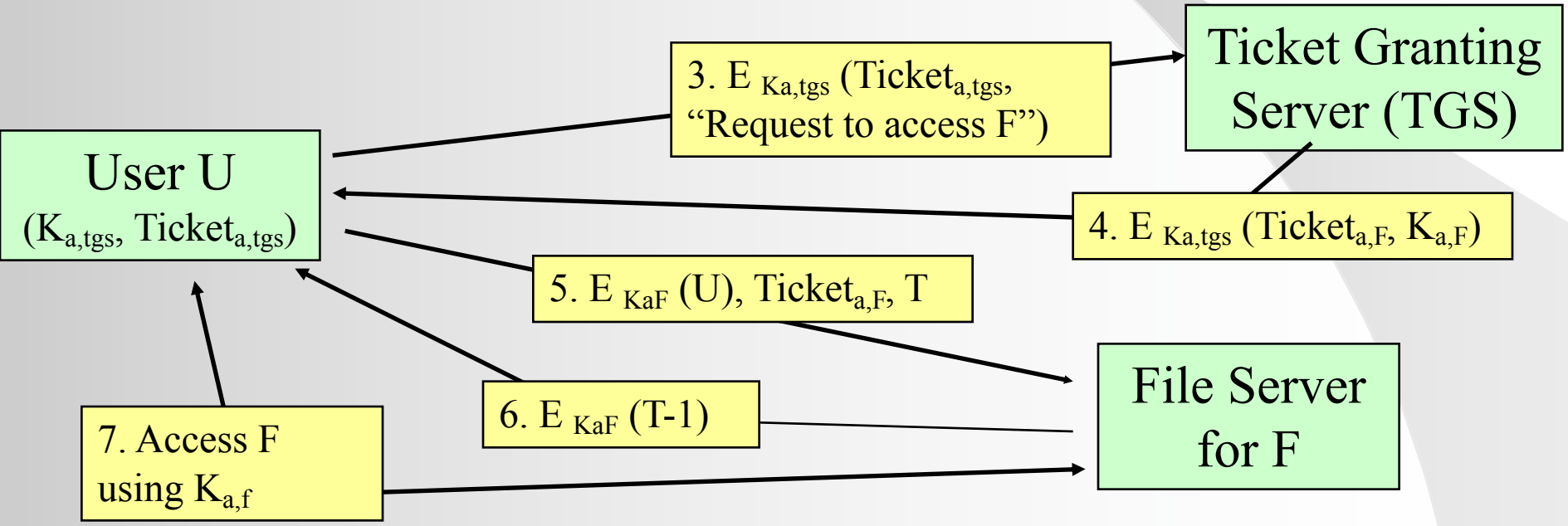


Kerberos Authentication Protocol
Initiating a Kerberos Session

- User's workstation sends the user's identity U to the Kerberos server when the user logs in
- The Kerberos server verifies that the user is authorized
- The Kerberos server sends 2 messages
  - To user's workstation, a session key $K_{a,tgs}$ for use in communication with the TGS, and a ticket $Ticket_{a,tgs}$ for TGS, $K_{a,tgs}$ is encrypted using the user's password pw (U can get the ticket with his password pw)
  - To the TGS, a copy of the session key $K_{a,tgs}$ and the identity of user U, encrypted with the key shared between Kerberos Server and the TGS K_{tgs}

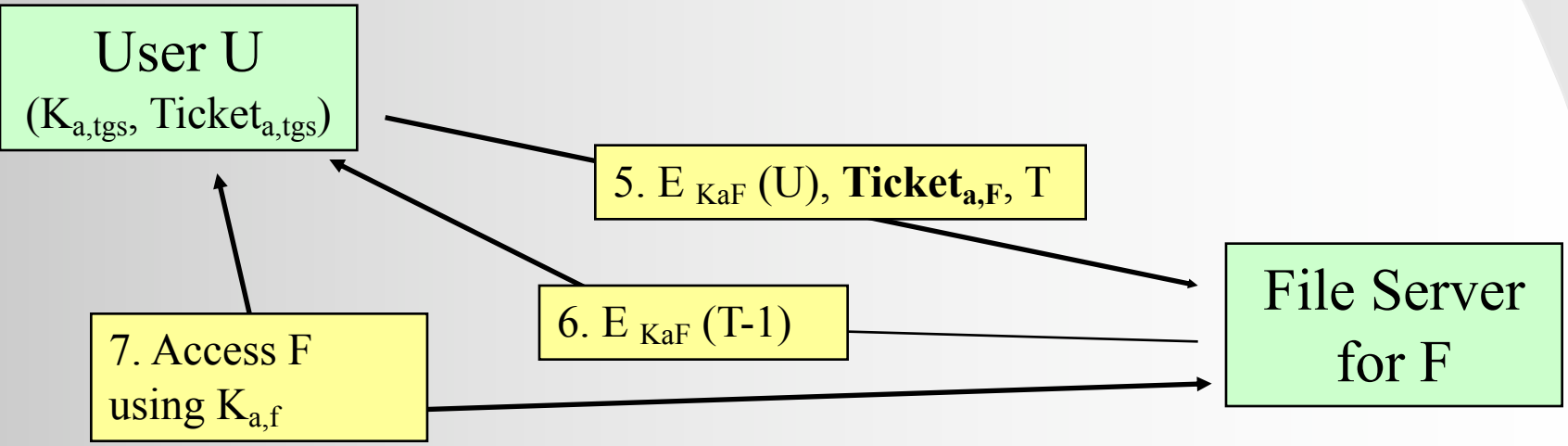
Kerberos Authentication Protocol Access a File

- User U wants to access a file F in File Server



What is in $Ticket_{a,F}$?

- User U wants to access a file F
- U requests a ticket to access F from TGS, and the request is encrypted using session key $K_{a,tgs}$
- TGS verifies U's access permission, it returns a ticket $Ticket_{a,F}$ and a session key $K_{a,F}$
- $Ticket_{a,F}$ encrypted by $K_{F,tgs}$, contains
  - U's authenticated identity
  - An identification of F
  - Access rights of F with respect to U
  - The session key $K_{a,F}$ for communication between file server and U
  - An expiration date for the ticket



Analyzing Kerberos

- No password communicated on the network
- Protected against spoofing using symmetric key encryption
- Limited period of validity for each ticket: protect against brute force search
- Timestamps to prevent replay attack
- Mutual authentication
  - Server authentication: only the file server F can decrypted the ticket issued by TGS which is encrypted by $K_{F,tgs}$ and then extracts the session key $K_{a,F}$ to be shared
  - How about client authentication?

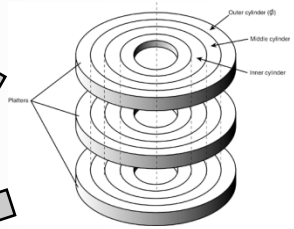
Some issues:

- Requires continuous availability of a trusted TGS
- Requires a trusted relationship between TGS and every server
- Requires timely transactions
- Password guessing works

Windows Login

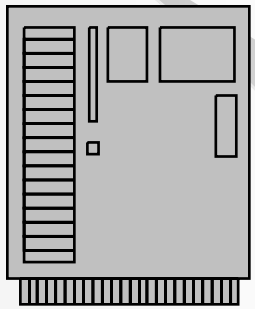
SAM DB

2. Authentication check



User U with password pw

1. (U,pw)



5. SAT created for U with SIDs and privileges

3. User U's SID and group SID

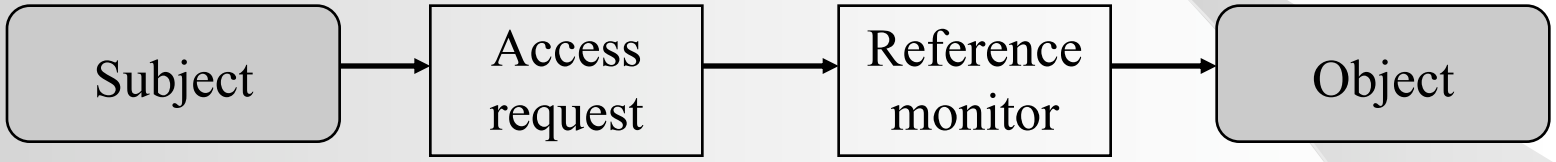
4. Create login session

Windows Login

- Users are prompted for username and password
- Username and password are gathered by the login process and passed on to the Local Security Authority (LSA) module
- The LSA calls the *authentication package* that compares the username and password against the values stored in SAM
- When a match is found, the SAM returns the user's security ID (SID) and the security ID of any group the user belongs to
- The *authentication package* creates a login session and passes this session together with all SIDs back to the LSA
- The LSA creates a *system access token* (SAT), containing the user's SIDs and user rights (privileges)
  - The access token contains all relevant information about the capabilities of the authenticated user
  - When deciding whether the user is allowed for a particular access, the access token will be consulted

Access Control

- Fundamental model of access control (Lampson 1982)
  - An active *subject* accessing a passive *object* with some specific *access operation*, while a *reference monitor* grants or denies access



- Example:
  - Subject: users or processes
  - Object: files or resources like memory, printer
- How to specify the control? 2 approaches:
  - What a subject is allowed to do: used in typical application system, e.g. user A is allowed to access database table T
  - What may be done with an object: used by typical OS, e.g. file gcc is executable and accessible by everyone

How to specify access rights?

- Access rights (Bell-LaPadula security model):
  - 4 access rights: execute, read, append (blind write), write
- How to specify?
  - Access control matrix
  - Capabilities
  - Access control lists
  - Group
  - Role-based access control

Access Control Matrix

- Access rights are defined in the form of an access control matrix (table)
- Each entry specifies the set of access operations subject *s* may perform on object *o*
- Not suitable for direct implementation if the number of subjects and objects is large or if the sets of objects and subjects change frequently
- E.g.

	Bill.doc	Edit.exe	Fun.com
Alice	-	{execute}	{execute, read}
Bill	{read, write}	{execute}	{execute, read, write}

Capabilities

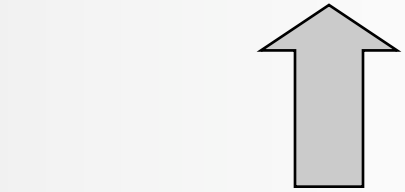
- One option is to implement the access control matrix: access rights are kept with subjects, called capabilities
- E.g.

Alice's capability	Edit.exe: execute; Fun.com: execute, read;
Bill's capability	Bill.doc: read, write; Edit.exe: execute; Fun.com: execute, read, write;

- When a subject creates a new object, it can give other subjects access to this object by granting them the appropriate capabilities
- When a subject calls another subject, it can pass on its capability

Problems with Capabilities

- Not widely used:
  - Difficult to get an overview of who has permission to access a given object, e.g. who has read access to Fun.com
  - Difficult to revoke a capability: either the OS has to be given the task or users have to keep track of all the capabilities they passed on
- May be useful when user carries their security policy during *roaming*, e.g. process A run by user U on machine M1 invokes an operation O on machine M2, what are the access rights for O on M2?



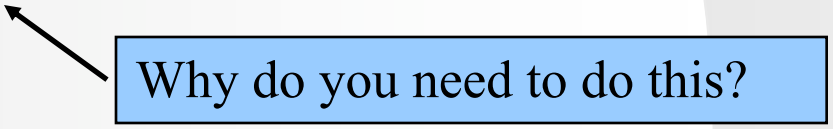
Any potential application?

Access Control Lists

- Another option is to implement the access control matrix: stores the access rights to an object with the object itself in an access control list (ACL)

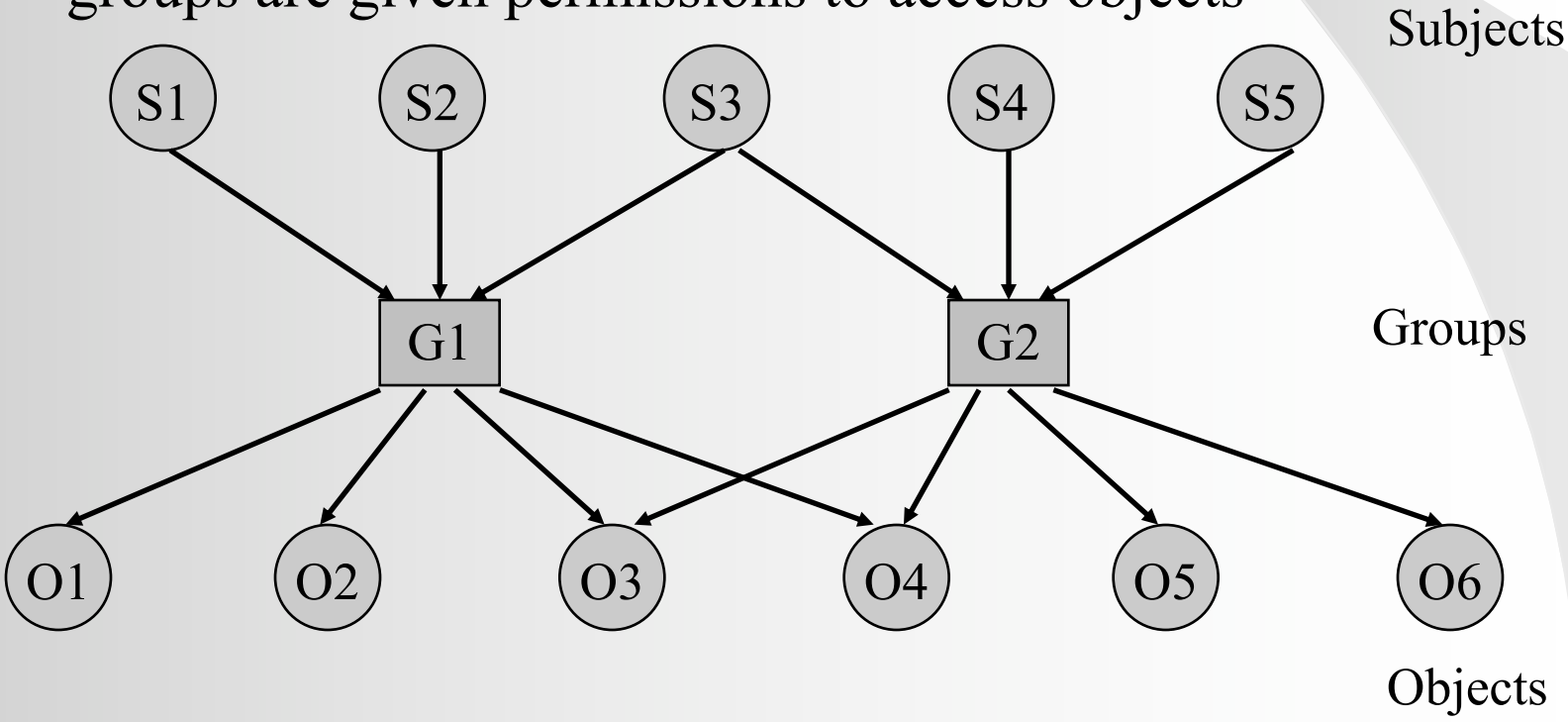
ACL for bill.doc	Bill: read, write
ACL for edit.exe	Alice: execute; Bill: execute
ACL for fun.com	Alice: execute, read Bill: execute, read, write

- ACLs are used for OSs that are geared towards managing access to objects
- To get an overview of permissions given to an individual user, you have to search through all ACLs



Intermediate Control using Group

- Managing a security policy expressed using access control matrix is a complex task in large systems, e.g. 1000 users with 10000 objects
- Users with similar access rights are collected in groups and groups are given permissions to access objects



Role-based Access Control

- Role-based access control (RBAC) focuses on the users and on the jobs users perform
- Subjects derive their access rights from the role they are performing
- Objects: Each object can be accessed only through the **procedures** defined for the object
- **Roles**:
  - A role is a collection of application specific operations (procedures)
  - Roles are assigned to users
  - A user can have more than one role and more than one user can have the same role

Procedures:

- High-level access control methods with a more complex semantic than read or write
- Procedures can only be applied to objects of certain *objects*, e.g. transfer *money* between *bank accounts*

Role-based Access Control Problems

- RBAC reduces identity theft to role theft: any one object or subject can compromise an entire group
- Possible violation of security policy due to role inheritance, automatic assumption of rules, unrestricted grants of privileges
- Possible misconfiguration due to fuzzy nature of overlaps between roles:
  - assignment of users to multiple roles
  - assignment of objects to multiple object-access groups

Windows Access Control

- Domains
  - A domain is a collection of machines sharing a common user accounts database and security policy
  - The master copy of the user accounts DB for the domain is held on a server called the primary domain controller (PDC), copy is held on a backup domain controller (BDC)
- Workstation can maintain their own local account DB and be a member of a domain at the same time
  - User can be a local user or a global user
  - A user with a local and a global account will have 2 different security identifiers (2 SIDs)
  - Resources can be managed globally or locally, e.g. a local printer or a network printer

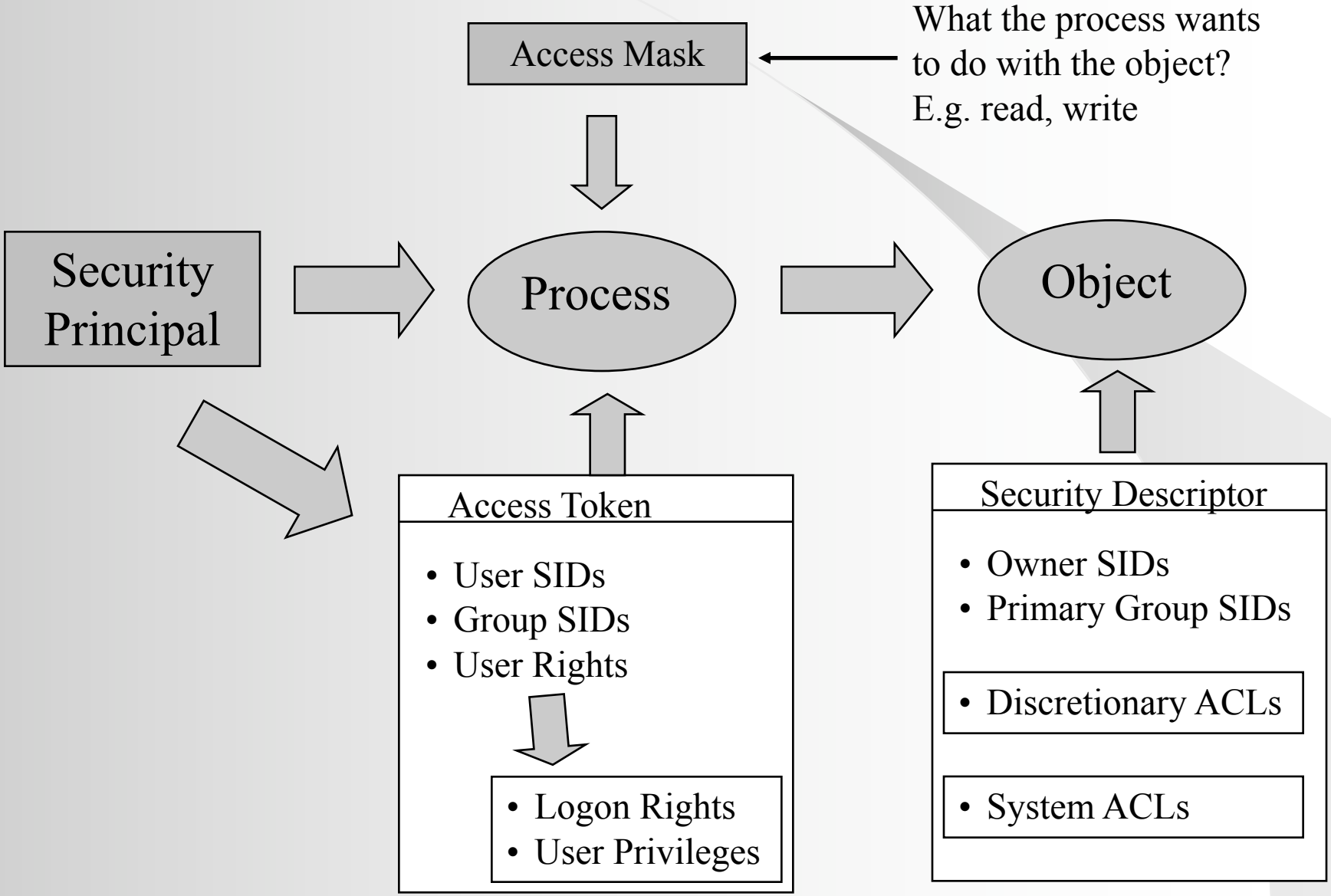
Windows Security Identifiers (SIDs)

- Every user, group and machine account has a unique security identification number (SID), used for discretionary access control (DAC)
- SID
  - Constructed when the account is created
  - Fixed for the life time of the account
  - Pseudo-random inputs are used in its construction, i.e. delete an account and recreate it will not have the same SID, and access permissions will be different
- When a domain is created, a unique SID is constructed
  - When a workstation or a server joins a domain, it receives a SID that includes the domain's SID
  - Copying the root directory and configuration files of one machine to another machine will make the 2 machines have identical SIDs, which violates the Windows security policy

What is inside SID?

- S-R-I-S-S- … -RID
  - R: revision number (currently 1)
  - I: identifier authority (48-bit)
  - S: 1-14 subauthority fields (32-bit)
  - RID: relative 32-bit identifier, unique in authority's name space
- Typical principals with their SID
  - Everyone (World): S-1-1-0
  - SYSTEM: S-1-5-18: the Windows on a machine runs locally as S-1-5-18
  - Administrator: S-1-5-21-<local authority>-500: a user account created during Windows installation
  - Administrators: S-1-5-32-544: built-in group with administrator privileges
  - Guest: S-15-21-<authority>-501

Windows Authorization Model



Access to Windows Objects

- Each object has a security descriptor:
  - The security ID of the owner of the object and the primary group SID
  - A set of discretionary access control lists (ACLs): contains access control permissions
  - A set of system access control list: contains auditing permissions, which control the audit messages to be generated
- An ACL is composed of multiple Access Control Entries (ACEs), each ACE is a permission
- An access control list entry (ACE) for a subject or group can be
  - AccessDenied: always listed first in an ACL
  - AccessAllowed: a list of access permission
  - E.g. the ACEs for a file exam.doc

Allow/Deny	**SID**	**Permission**
AccessAllowed	KPC	Read, Write
AccessDenied	KMC	
AccessDenied	Pierre	

Windows Access Control Checking

- When a subject requests access to an object, the security reference monitor takes the subject's security access token (SAT) which contains the subject's SID and privileges, and the object's ACL to determine whether the requested access should be granted:

1. If no ACL exists, no checks are performed and access is granted
2. If an ACL exists, then for each ACE, the subject's SID in the SAT is compared with the SIDs in the ACE:
   a) The ACE does not contain a matching SID, the ACE is skipped
   b) The ACE contains a matching ID specifying 'AccessDenied', no access is permitted, regardless of any conflicting AccessAllowed flags
   c) The ACE contains a matching SID specifying 'AccessAllowed', access is granted
3. If no matching entry is found, access is denied

Example: Windows Access Control Checking

login

User1 with SID1

run

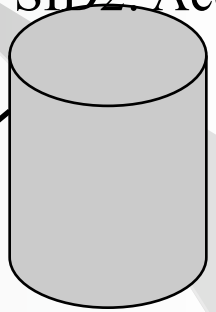
Program1 with SID1

F3's ACL

SID2: AccessDenied

read file F3

OK or fail?

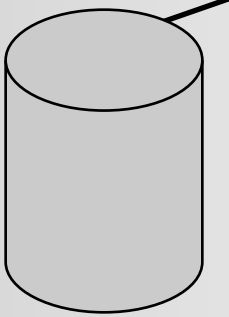


F2's ACL

SID1: AccessDenied

read file F2

FAIL

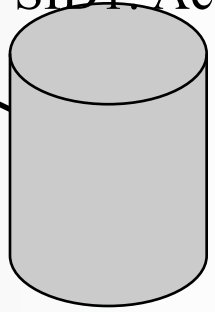


F1's ACL

SID1: AccessAllowed

read file F1

OK



Accountability

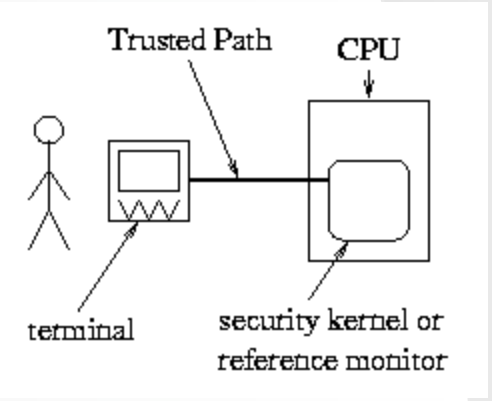
- Accountability requirements are the second set of security requirements defined in the Orange book (Department of Defense Trusted Computer Evaluation Criteria, DoD 5200.28-STD, Dec 1985)
- Accountability: the system knows who you are and what you're doing
  - The system must be able to identify all users of the system
  - The system must use information about your identity to decide whether you can access certain information
  - The system must keep track of any security-related actions you take
- Secure system uses your ID to maintain individual accountability, i.e. keep track of what you are doing in a system, e.g.
  - If KPC repeatedly tries to access files he's not authorized to view, the system will know

Accountability Requirements (1)

- Defined in Orange Book:
  - Identification and authentication
  - Trusted path
  - Audit
- Identification and authentication
  - Requires you to identify yourself to the system before performing any work that will require interaction with the TCB (Trusted Computing Base), e.g. running a program, reading a file, …
  - TCB refers to mechanisms that enforce security in a system

Accountability Requirements (2)

- **Trusted path**:
  - A trusted path provides an unmistakable means by which a user (at a terminal or a workstation) can communicate directly with the secure system without having to interact with the system through untrusted applications
  - A trusted path is the *flip* side of the identification and authentication requirement, i.e. authenticate the secure system
  - E.g. use a particular key sequence on a terminal that signals the secure system to halt any process that is running on the terminal and to establish the direct link to the TCB, like the <BREAK> key followed by <RETURN> key in VAX/VMS



Accountability Requirements (3)

- Audit
  - The recording, examining and reviewing of security related activities in a trusted system, i.e. activities that related to a subject's access to an object
  - Typical events are:
    - Logons (successful or unsuccessful)
    - Logouts
    - Remote system accesses
    - File opens, closes, renames, and deletions
    - Changes in privileges or security attributes
  - Auditing provides a way of determining whether and how an attack may take place