Protection and Security

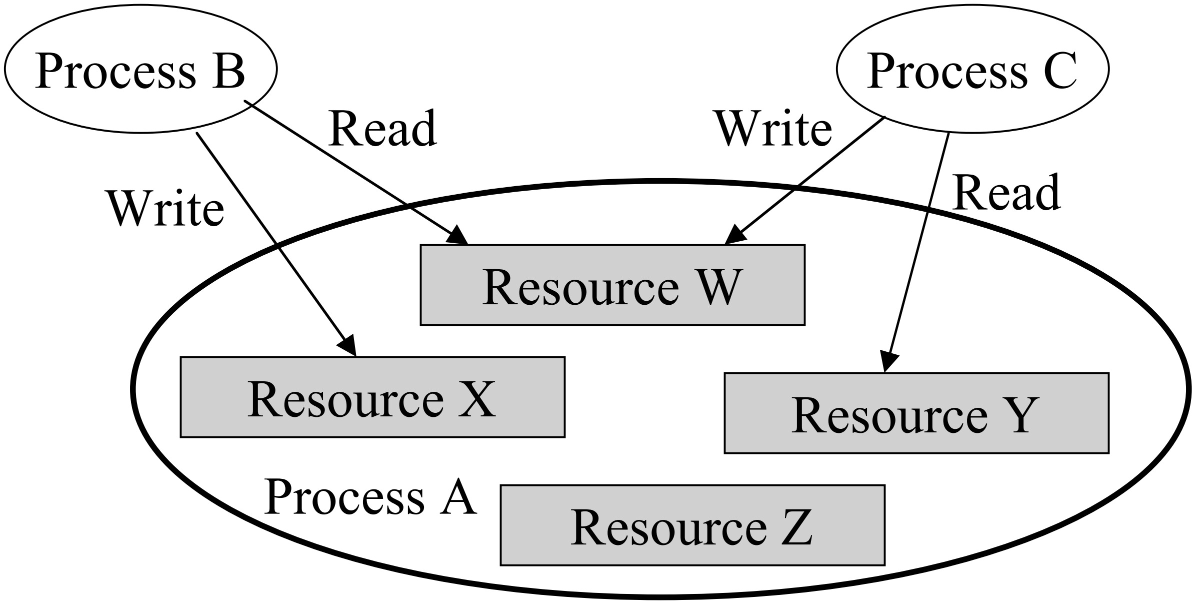
Policy & Mechanism

- *Protection mechanisms* are used to authenticate access to resources
  - File protection
  - Memory protection
- A *security policy* reflects an organizations strategy to authorize access to the computer’s resources
  - Managers have access to personnel files
  - OS processes have access to the page table

Authentication

- User/process authentication
  - Is this user/process who it claims to be?
    - Passwords
    - More sophisticated mechanisms
- Authentication in networks
  - Is this computer who it claims to be?
    - File downloading
    - Obtaining network services
    - The Java promise

Internal Access Authentication

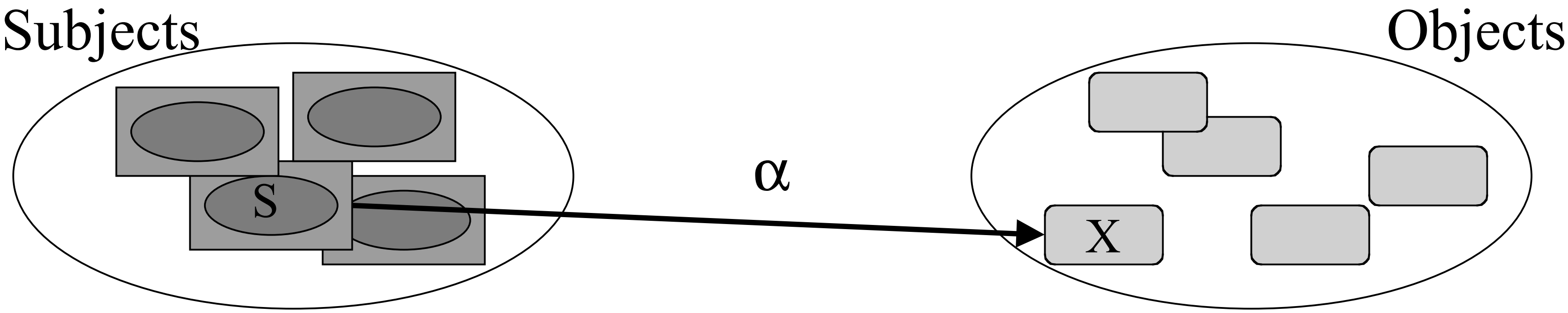


- Sharing parameters
- Confinement
- Allocating rights
- Trojan horse

Lampson’s Protection Model

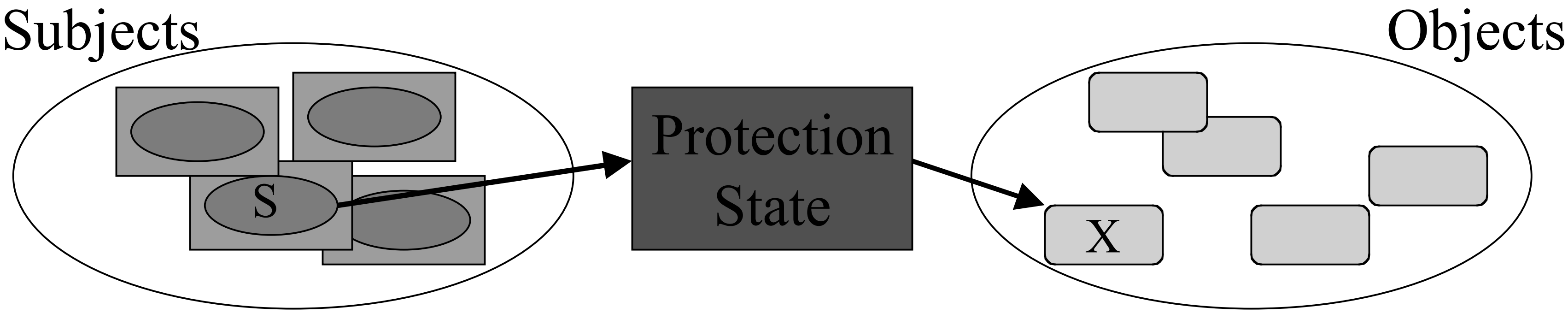
- Active parts (e.g., processes)
  - Operate in different domains
  - *Subject* is a process in a domain
- Passive parts are called *objects*
- Want mechanism to implement different security policies for subjects to access objects
  - Many different policies must be possible
  - Policy may change over time

A Protection System



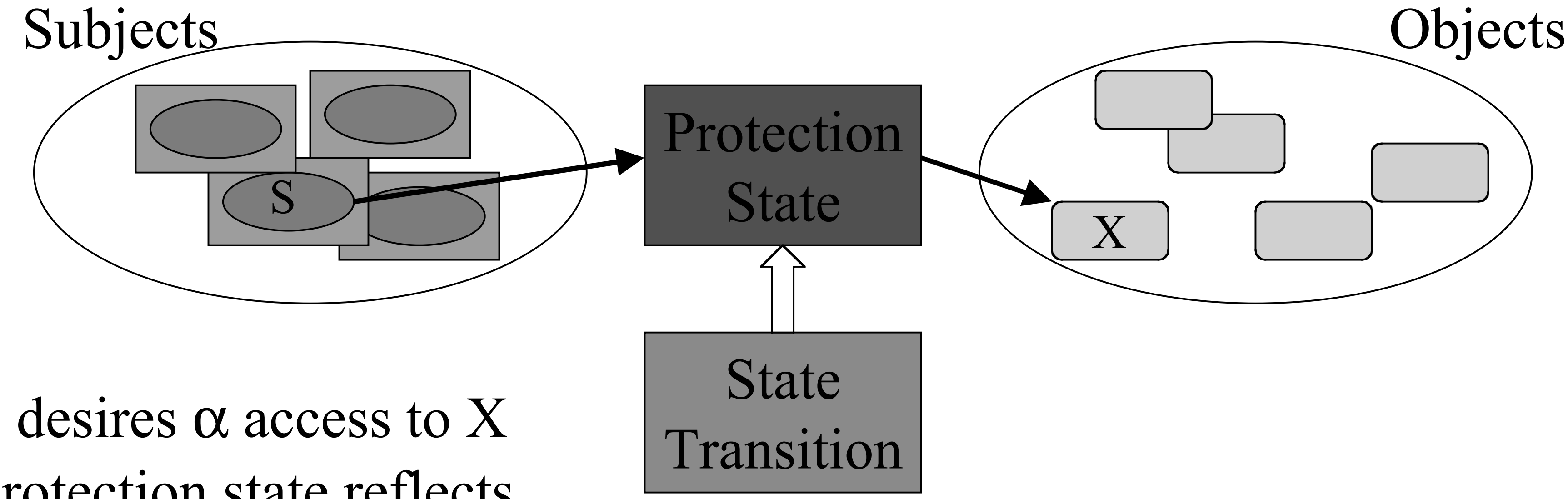
- S desires α access to X

A Protection System



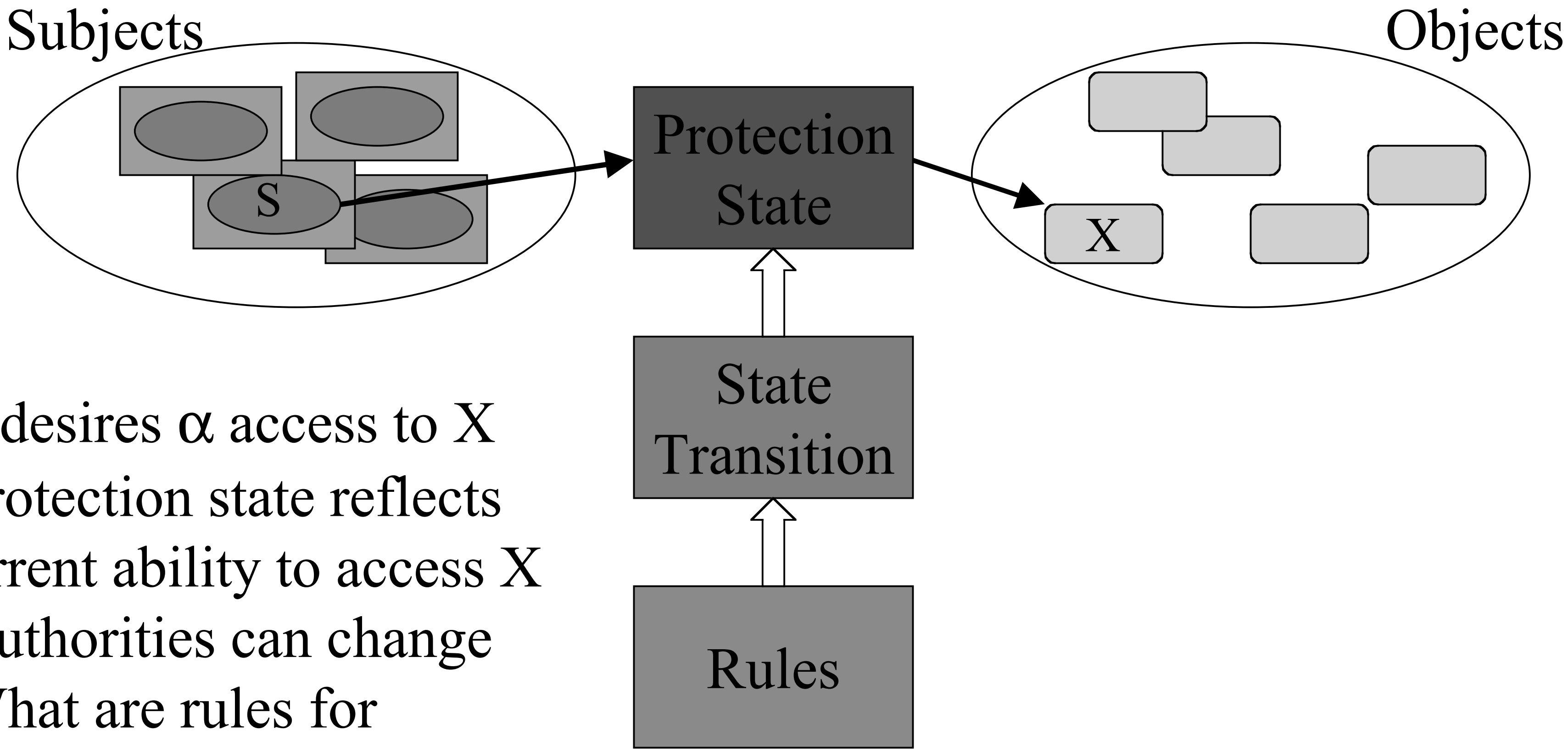
- •S desires α access to X
- •Protection state reflects current ability to access X

A Protection System



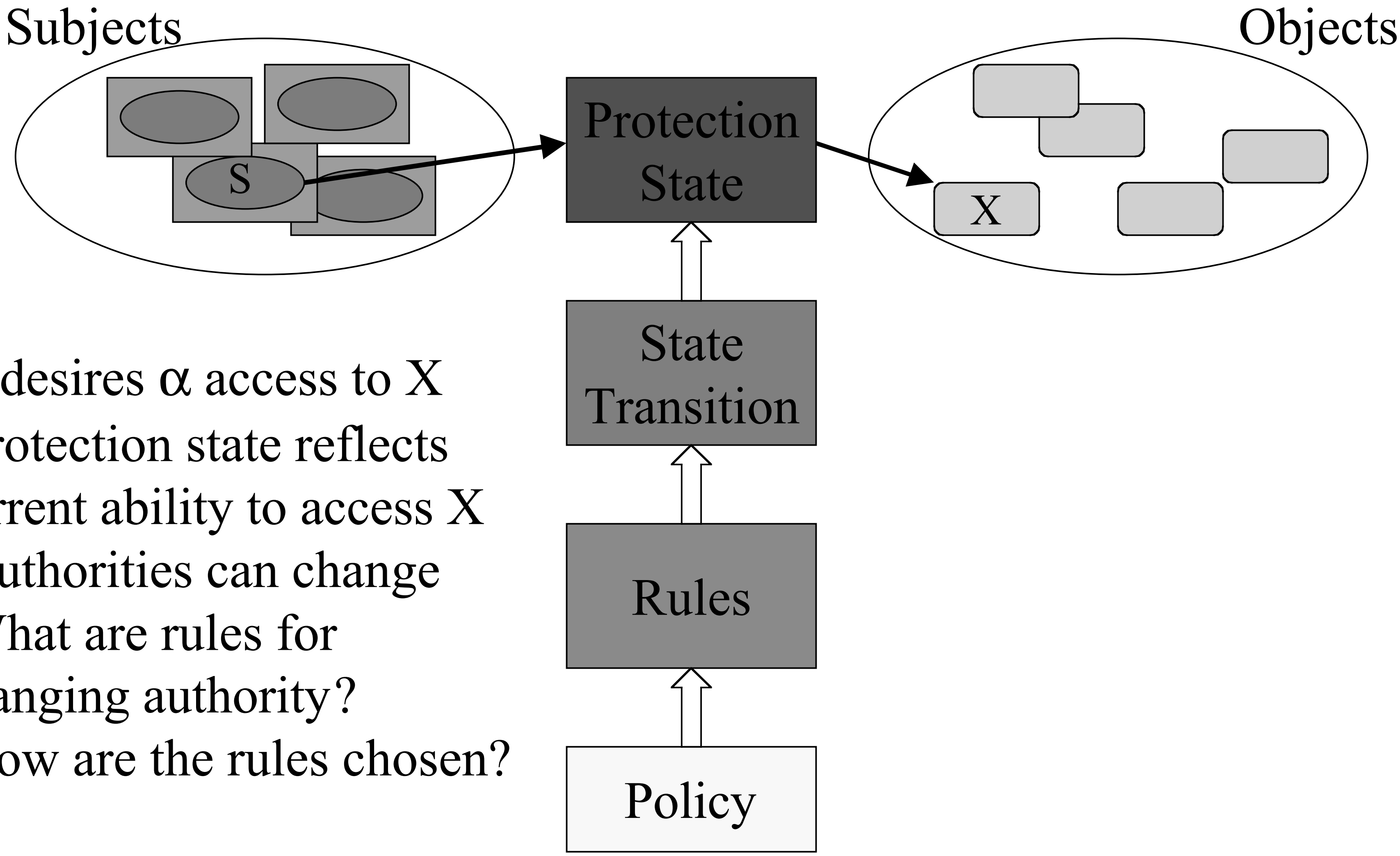
- S desires α access to X
- Protection state reflects current ability to access X
- Authorities can change

A Protection System



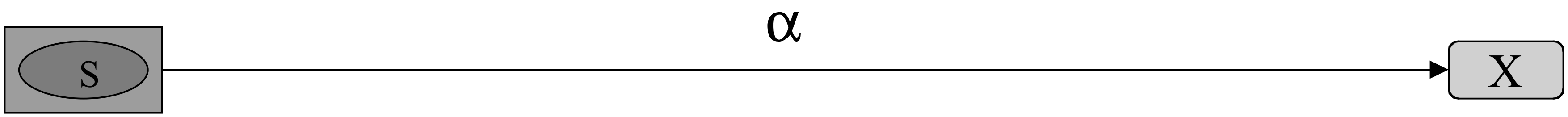
- •S desires α access to X
- •Protection state reflects current ability to access X
- •Authorities can change
- •What are rules for changing authority?

A Protection System



- S desires α access to X
- Protection state reflects current ability to access X
- Authorities can change
- What are rules for changing authority?
- How are the rules chosen?

Protection System Example



- S desires α access to X

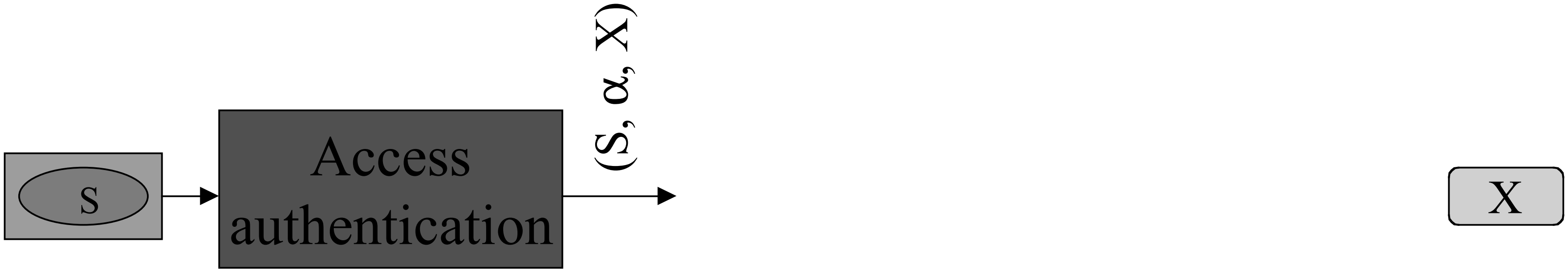
Protection System Example

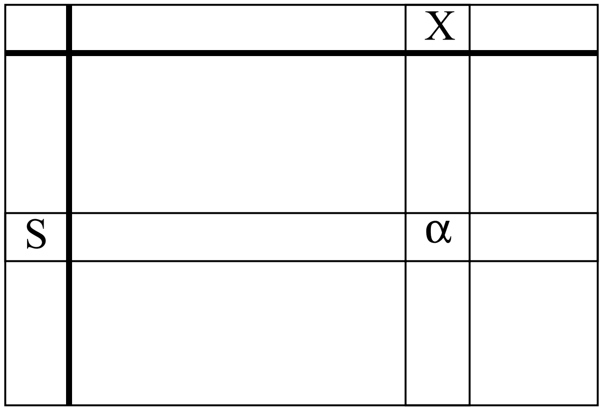
- S desires α access to X
- Captures the protection state

		X	
S		α	

Access matrix

Protection System Example

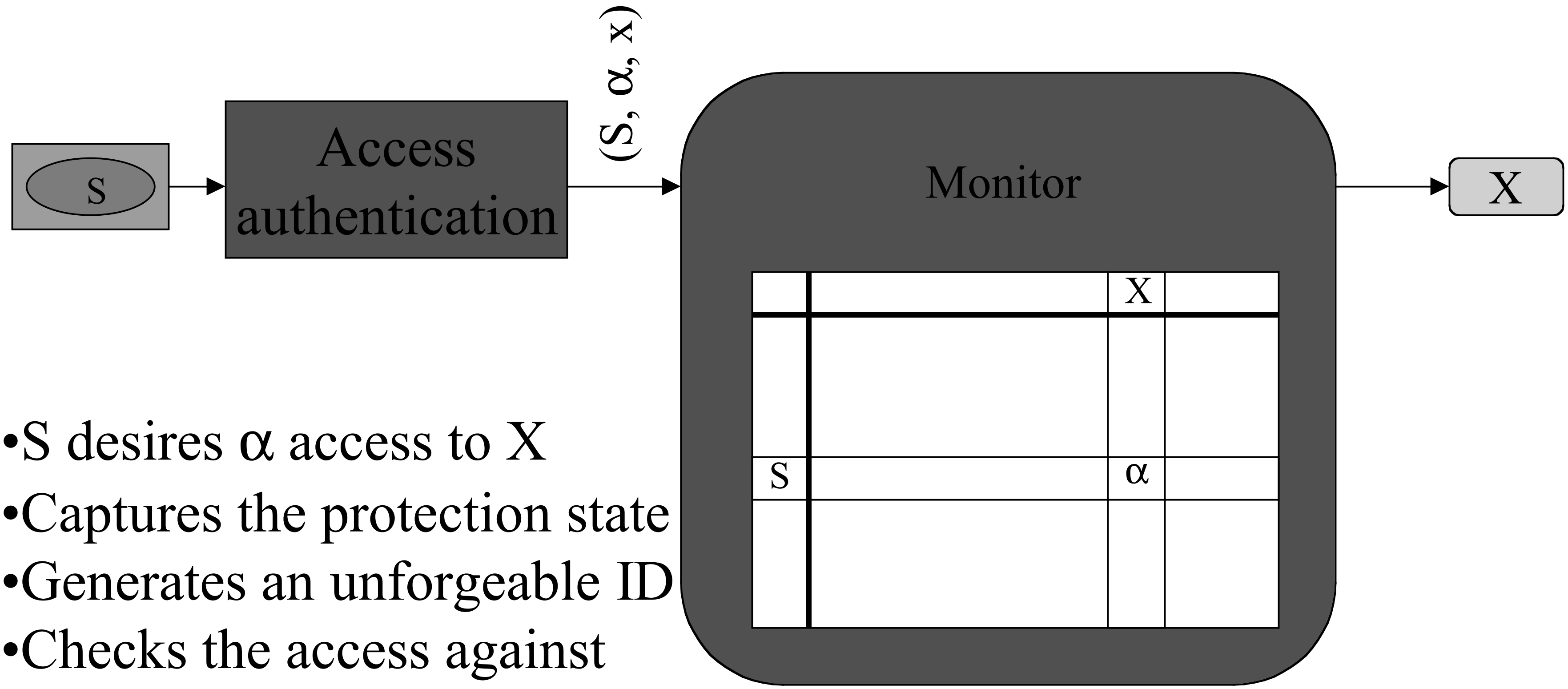




Access matrix

- S desires α access to X
- Captures the protection state
- Generates an unforgeable ID

Protection System Example

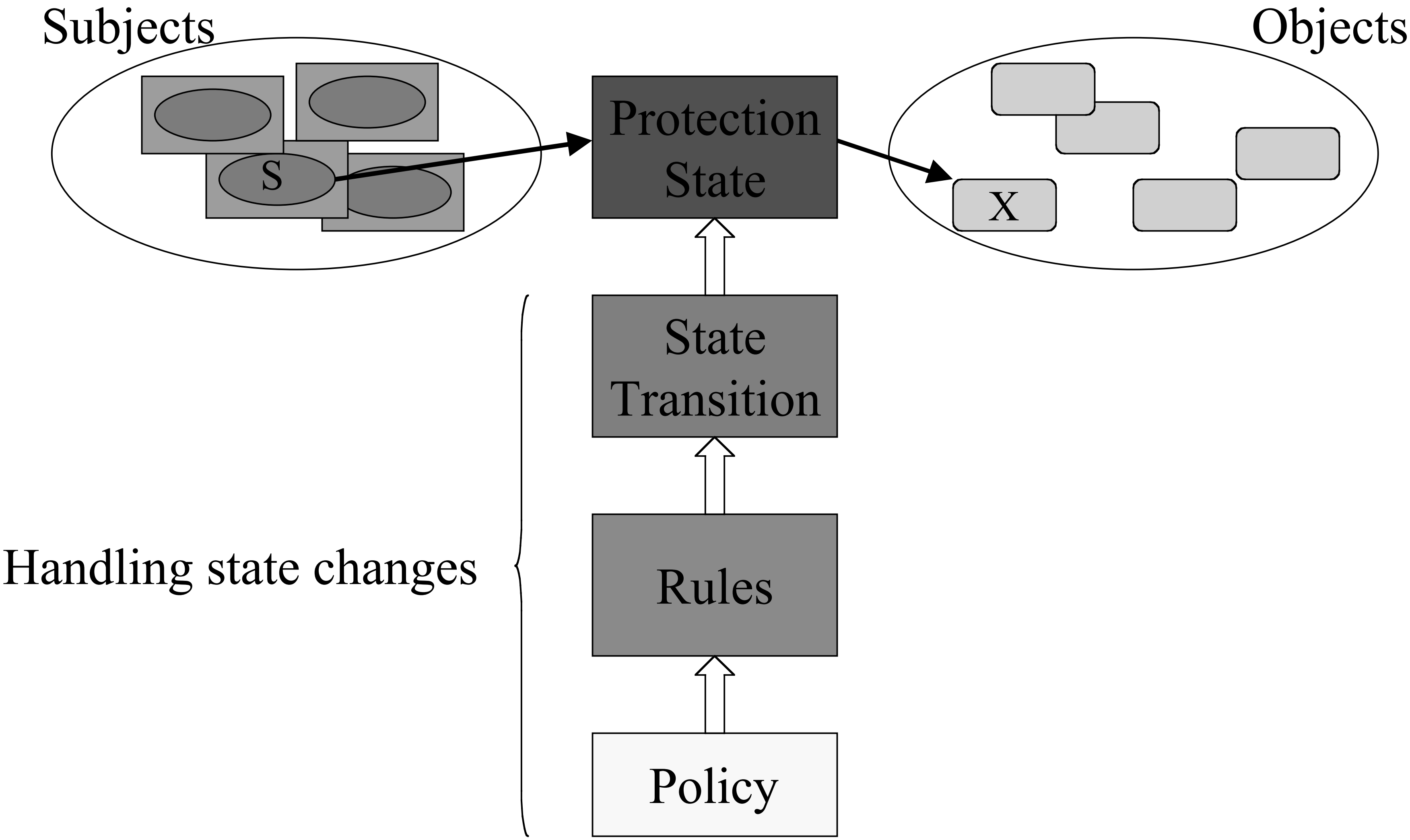


- S desires α access to X
- Captures the protection state
- Generates an unforgeable ID
- Checks the access against the protection state

Protection State Example

	S_1	S_2	S_3	F_1	F_2	D_1	D_2
S_1	control	block wakeup owner	control owner	read* write*		seek	owner
S_2		control	stop	owner	update	owner	seek*
S_3			control	delete	execute owner		

A Protection System



Policy Rules Example

	S_1	S_2	S_3	F_1	F_2	D_1	D_2
S_1	control	block wakeup owner	control owner	read* write*		seek	owner
S_2		control	stop	owner	update	owner	seek*
S_3			control	delete	execute owner		

Rules for a Particular Policy

Rule	**Command by S_0**	**Authorization**	**Effect**
1	transfer($\alpha\vert\alpha*$) to (S, X)	$\alpha* \in A[S_0, X]$	$A[S, X] = A[S, X] \cup \{\alpha\vert\alpha*\}$
2	grant($\alpha\vert\alpha*$) to (S, X)	owner $\in A[S_0, X]$	$A[S, X] = A[S, X] \cup \{\alpha\vert\alpha*\}$
3	delete α from (S, X)	control $\in A[S_0, S]$ or owner $\in A[S_0, X]$	$A[S, X] = A[S, X] - \{\alpha\}$

Protection Domains

- Lampson model uses processes and domains -- how is a domain implemented?
  - Supervisor/user hardware mode bit
  - Software extensions -- *rings*
- Inner rings have higher authority
  - Ring 0 corresponds to supervisor mode
  - Rings 1 to S have decreasing protection, and are used to implement the OS
  - Rings S+1 to N-1 have decreasing protection, and are used to implement applications

Protection Domains (cont)

- Ring crossing is a domain change
- Inner ring crossing ⇒ rights amplification
  - Specific *gates* for crossing
  - Protected by an authentication mechanism
- Outer ring crossing uses less-protected objects
  - No authentication
  - Need a return path
  - Used in Multics and Intel 80386 (& above) hardware

Implementing Access Matrix

- Usually a sparse matrix
  - – Too expensive to implement as a table
  - – Implement as a list of table entries
- Column oriented list is called an *access control list* (ACL)
  - – List kept at the object
  - – UNIX file protection bits are one example
- Row oriented list is a called a *capability list*
  - – List kept with the subject (i.e., process)
  - – Kerberos ticket is a capability
  - – Mach mailboxes protected with capabilities

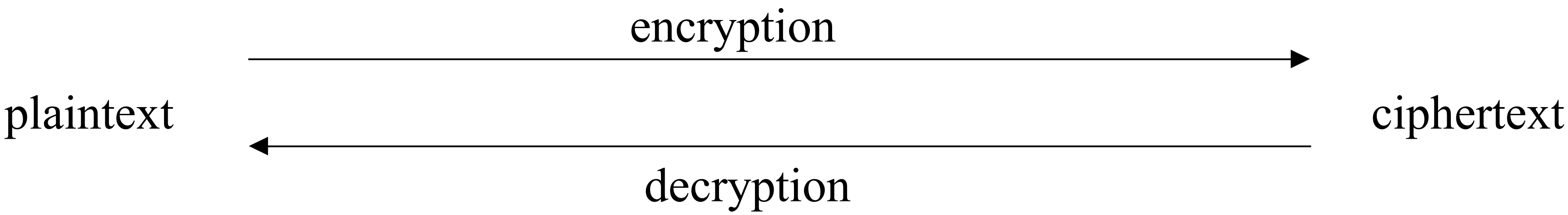
More on Capabilities

- Provides an address to object from a very large address space
- Possession of a capability represents authorization for access
- Implied properties:
  - Capabilities must be very difficult to guess
  - Capabilities must be unique and not reused
  - Capabilities must be distinguishable from randomly generated bit patterns

Cryptography

- Information can be encoded using a *key* when it is written (or transferred) -- *encryption*
- It is then decoded using a key when it is read (or received) -- *decryption*
- Very widely used for secure network transmission

More on Cryptography



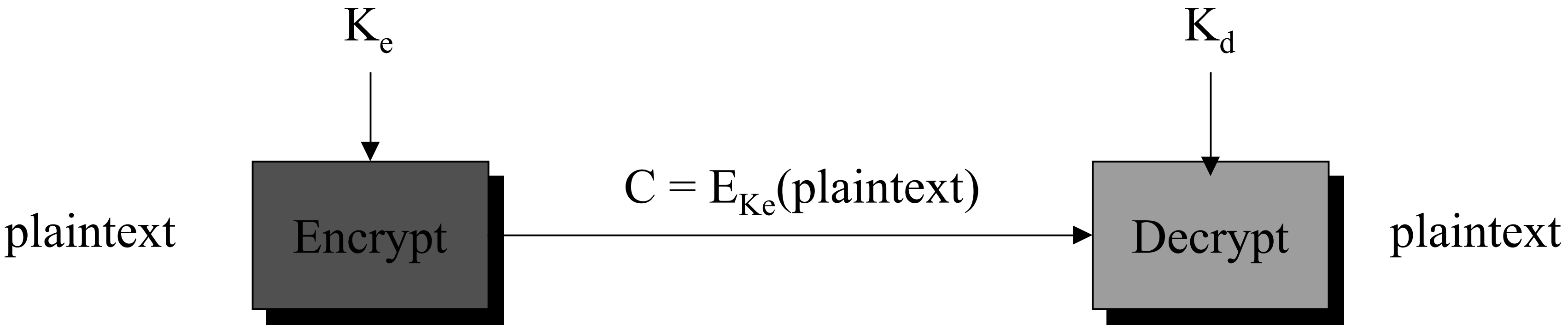
encryption

plaintext

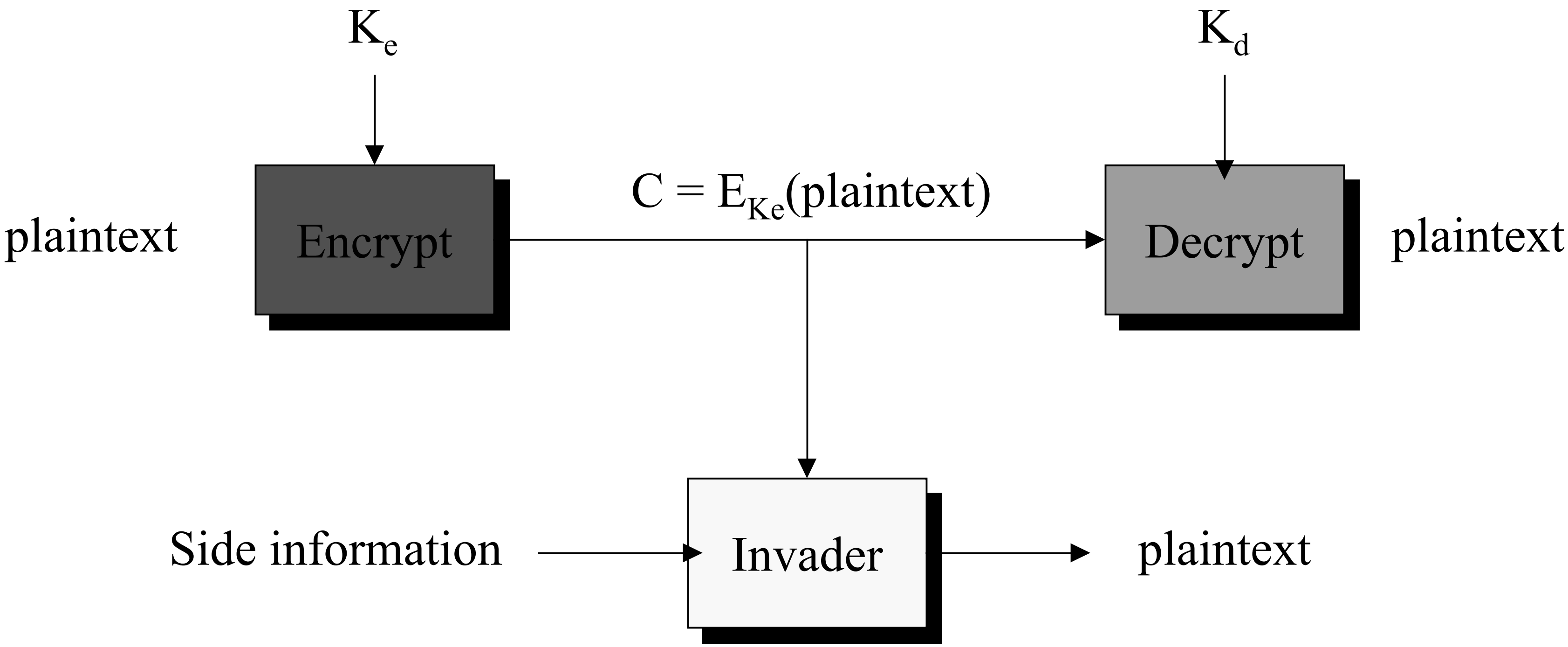
ciphertext

decryption

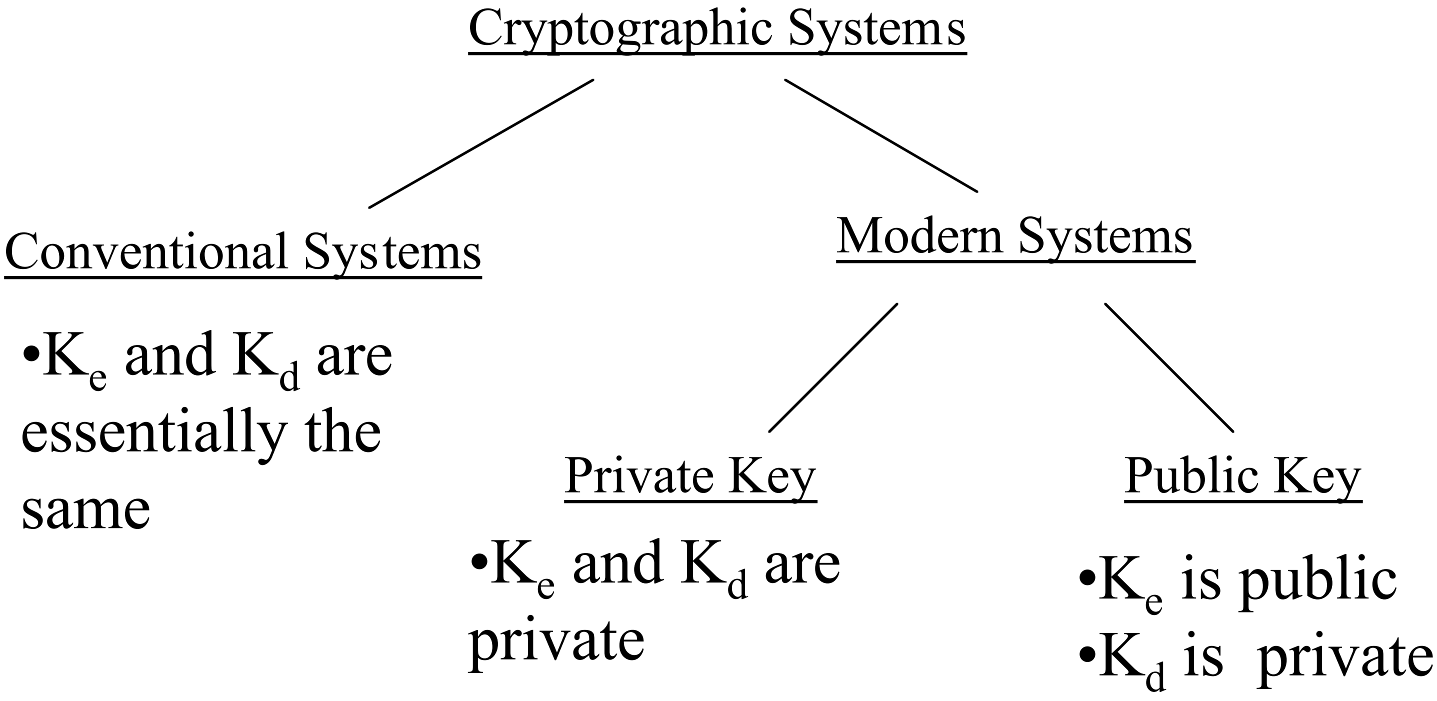
More on Cryptography



More on Cryptography



Cryptographic Systems



Cryptographic Systems

Conventional Systems

- K_e and K_d are essentially the same

Modern Systems

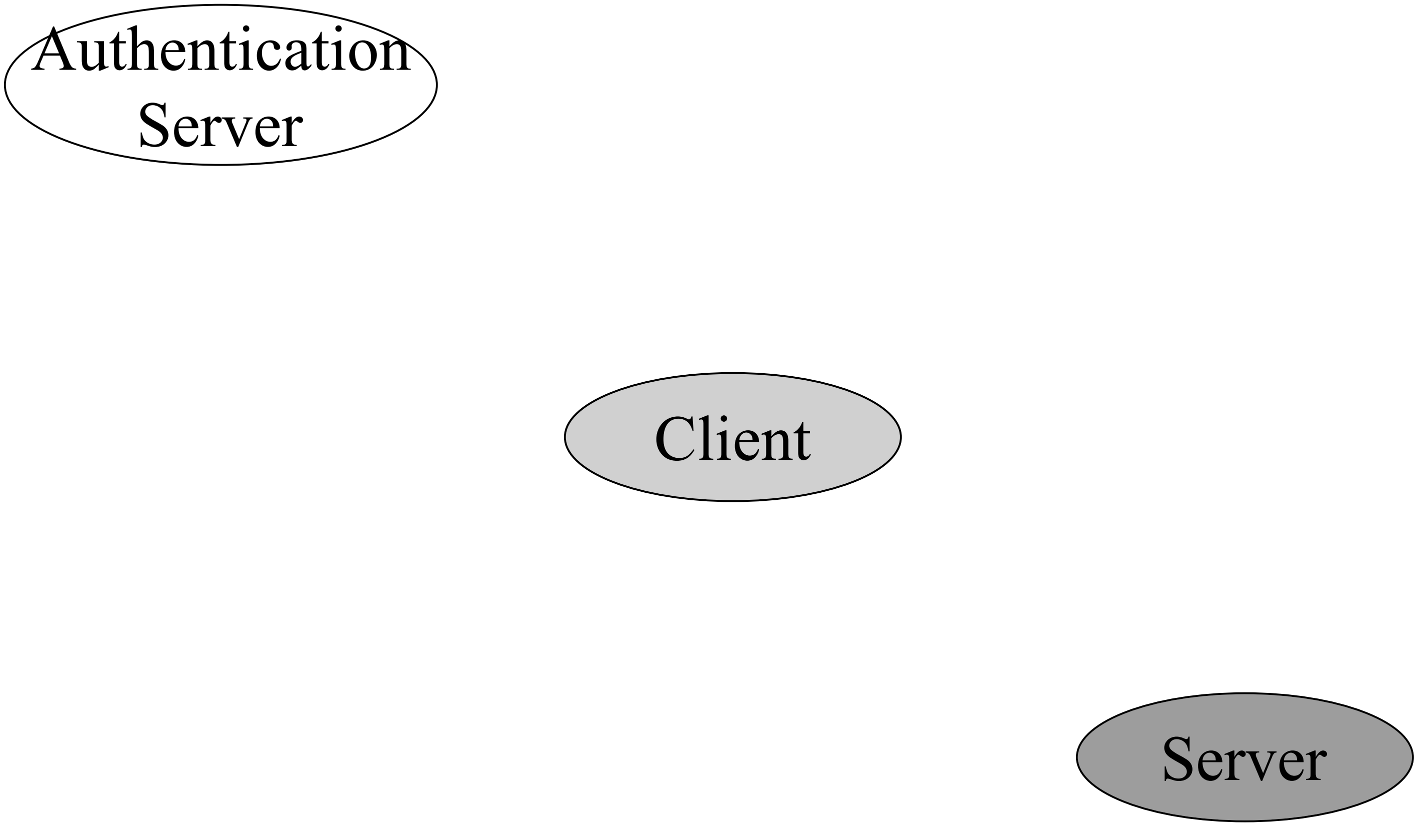
Private Key

- K_e and K_d are private

Public Key

- K_e is public
- K_d is private

Kerberos

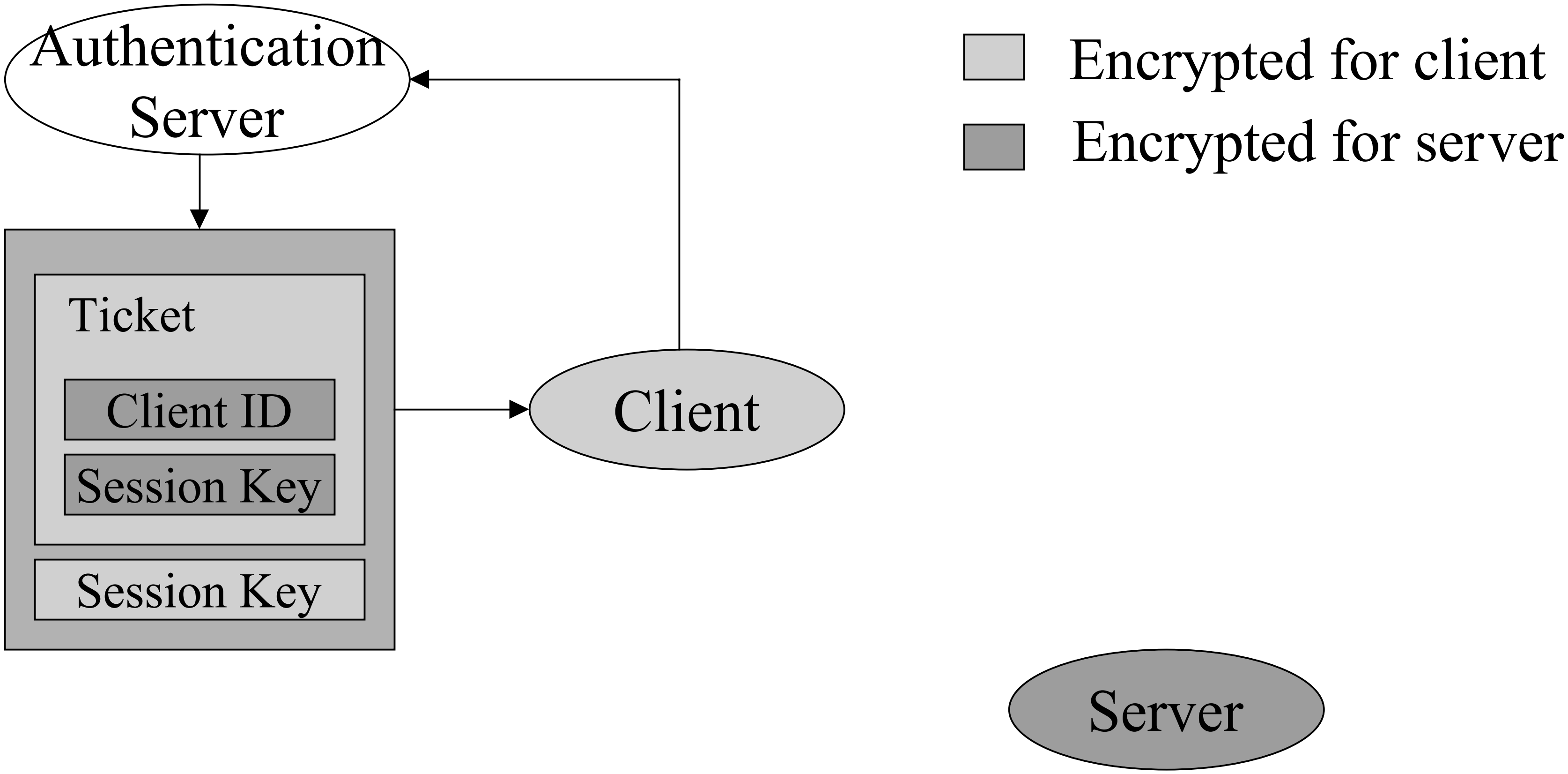


Authentication Server

Client

Server

Kerberos



Authentication Server

Ticket

Client ID

Session Key

Session Key

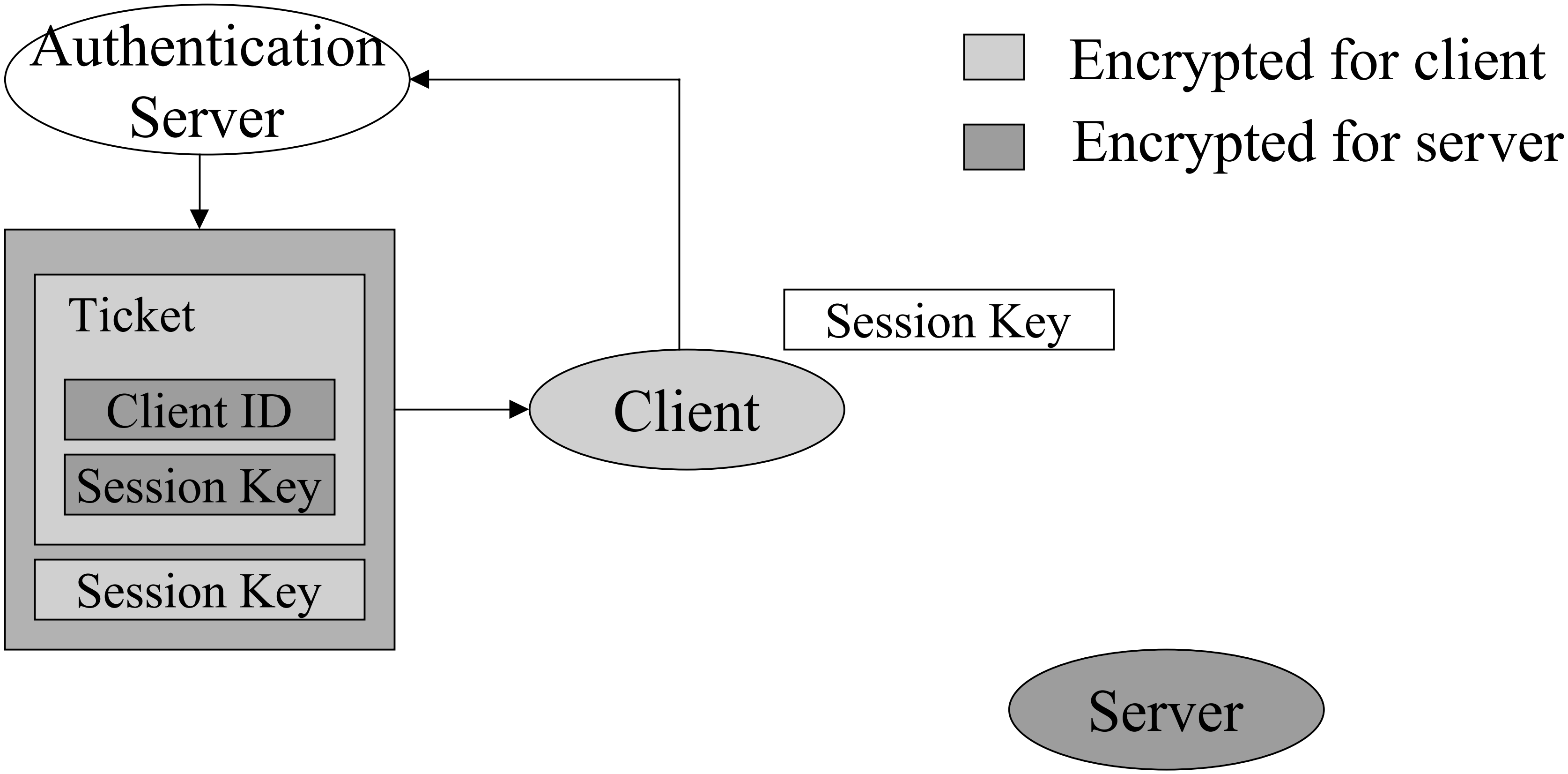
Client

Server

░ Encrypted for client

▓ Encrypted for server

Kerberos



Authentication Server

Ticket

Client ID

Session Key

Session Key

Client

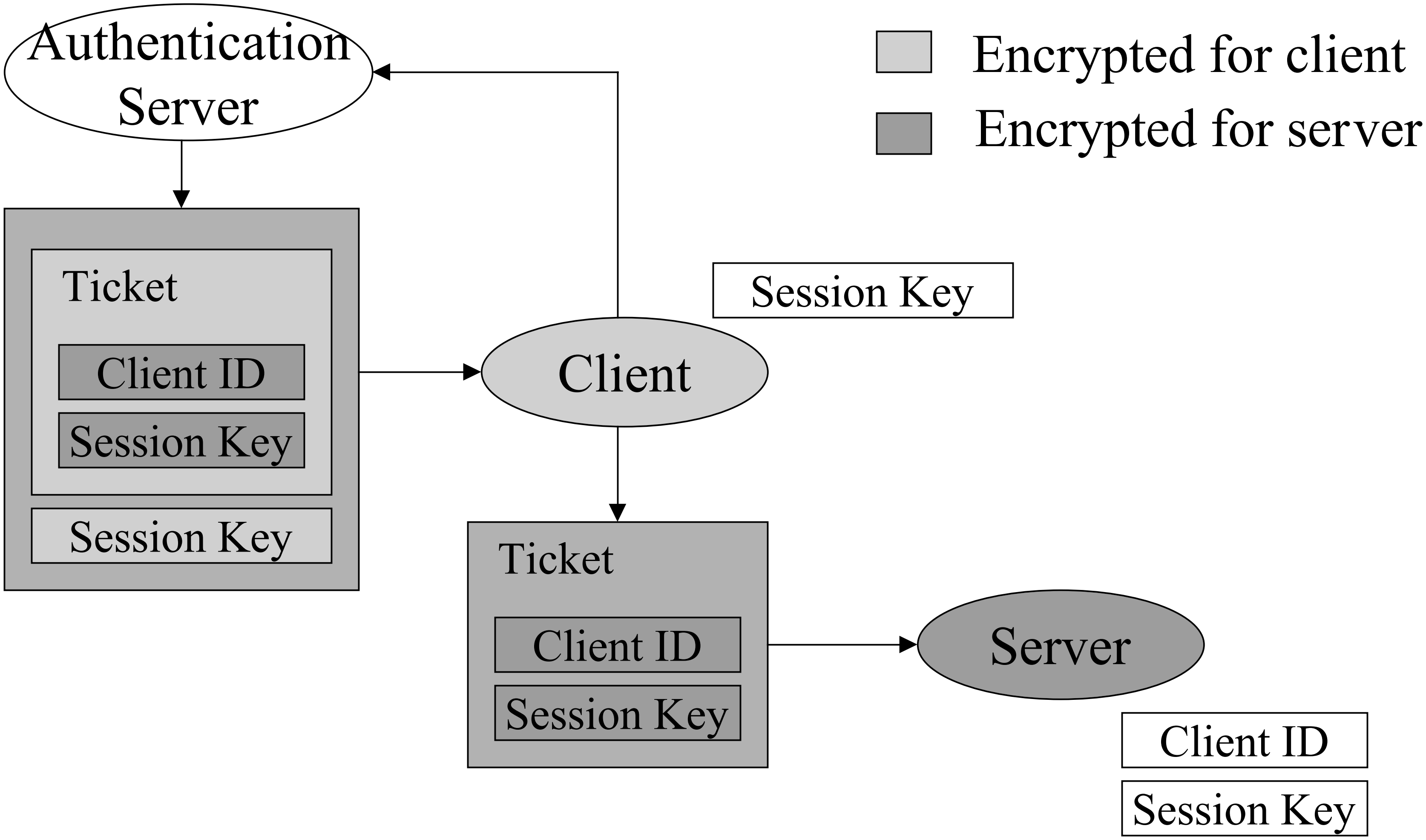
Session Key

Server

▨ Encrypted for client

▨ Encrypted for server

Kerberos



Authentication Server

Encrypted for client

Encrypted for server

Ticket

Client ID

Session Key

Session Key

Client

Session Key

Ticket

Client ID

Session Key

Server

Client ID

Session Key