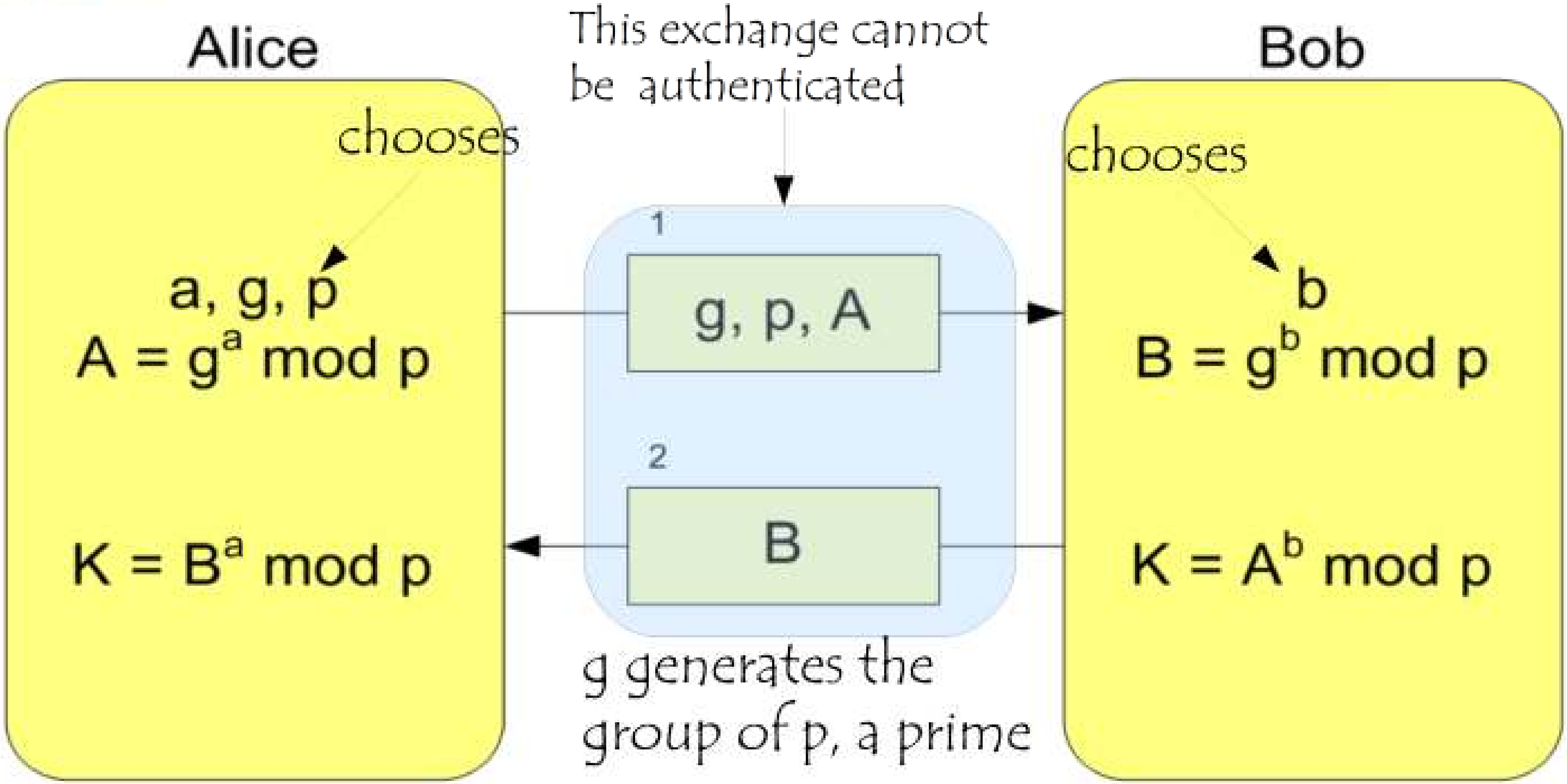
Diffie-Hellman



Alice

chooses

a, g, p

$A = g^a \bmod p$

$K = B^a \bmod p$

This exchange cannot be authenticated

1

g, p, A

2

B

g generates the group of p, a prime

Bob

chooses

b

$B = g^b \bmod p$

$K = A^b \bmod p$

$$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$$

VPN and symmetric encryption -II

- Each firewall pubblish a pubblic key and know the corresponding secret key
- The two keys makes it possible to compute a symmetric key for each ordered pair of firewalls
- Data to be exchanged is protected with the symmetric key
- IP v6

VPN: a shared problem

- Any implementation of any VPN may be the target of a Denial of Service attack
- A VPN has to decrypt any message it receives. If the output satisfies the protocol, it forwards the cleartext otherwise it discards the message
- On receiving a flood of fake messages, the receiver will be busy to discard them and cannot run legal applications or receive legal messages
- This shows that any security solutions that only applies encryption cannot guarantee resource availability

IPSEC

- An IPv4 extension to authenticate and encrypt information flows, to be used till IPv4 will be replaced by IPv6 ☺ ☺
- There are further solutions that offer security service at distinct level of the OSI stacks (PGP, HTTPS, SSL, etc).
- Two IPSEC behaviours (protocols)
  - Authentication Mode = authentication header
  - Encapsulated Security Payload = the information is encrypted
  - Both protocols can be used in one of two modes
    - Transport Mode = the original packet is updated by inserting new fields
    - Tunnel Mode = the old IP is protected and becomes the information of a new packet

IPSEC

- IPSEC can also be used between
  - two hosts (even clients),
  - a gateway and an host
  - two gateways.
- By replacing IP with IPSEC, we increase communication confidentiality and integrity in a more transparent way for the involved hosts
- No update to the software or hardware network components to adopt IPSEC.

IPSEC

IPSEC defines the following, further protocols

- **AH** (Authentication Header) it protect the integrity of and authenticate the data
- **ESP** (Encapsulating Security Payload) it offers confidentiality because of payload encryption.
- **IKE** (Internet Key Exchange) two partners can agree on the key to be used and on how long it should be used
- **ISAKMP** (Internet Security Association and Key Management Protocol) it is used to set up and update " **Security Association (SA)**" and their attributes

IPSEC-SA

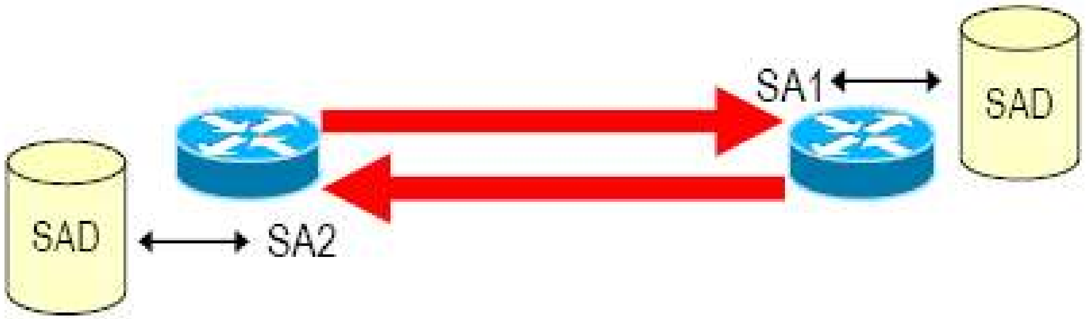
- A Security Association (SA) describes a directed connection together with the security services paired with the traffic it transmits

- To secure a bidirectional connection, two SAs are required, one in each direction

- An SA also includes any information to execute the security services

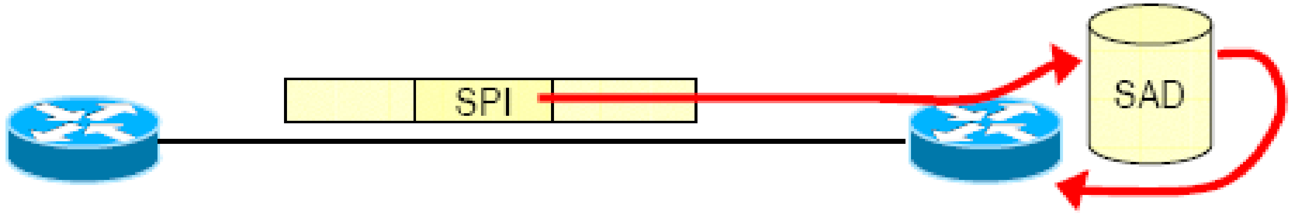
- The security services of an SA are implemented either through AH or through ESP. In general the protocols are never applied simultaneously

SA unidirectional



- ➔ **SPI = Security Parameters Index**
  - ⇨ The (somewhat) unique "name" of an SA
- ➔ **Destination Address based**
  - ⇨ Security Association managed at the receiver side
- ➔ **SAD = Security Associations Database**
  - ⇨ SPI = search key (at least)
  - ⇨ Stores set of security services per each SA, and related parameters
    - →E.g. which encryption algorithm; shared key for encryption, SA lifetime, Sequence number counter, etc

SPI – Header field



- ➔ **32 bit index**
- ➔ **Used to lookup the SAD at destination**
  - ⇨ Lookup also uses
    - →destination address
    - →source address
    - →security protocol (AH/ESP)
- ➔ **Retrieves algorithms and parameters that allow to process received packet**

IPSEC

There are two types of SA that introduce some updates to an IP packet:

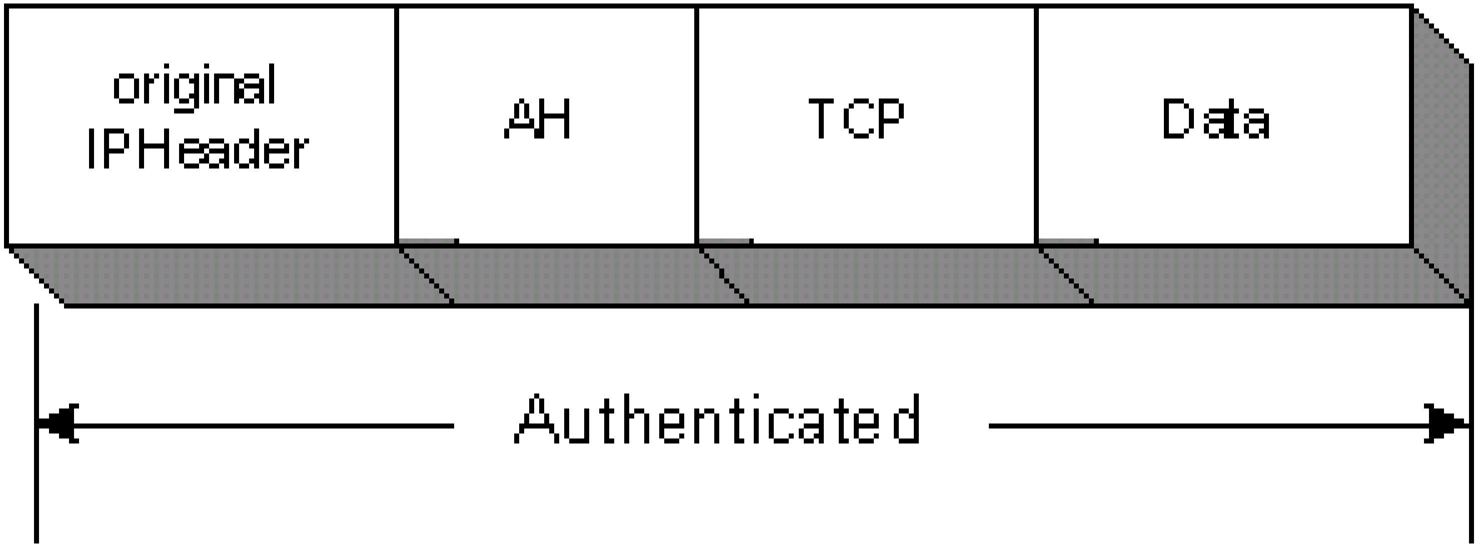
Transport mode (SA between two hosts) the security header immediately follows the IP header.

Tunnel mode (at least one end point is a gateway) there are two IP headers

- The first one is the more external one and it shows where the tunnel ends
- The inner one defines the packet final destination

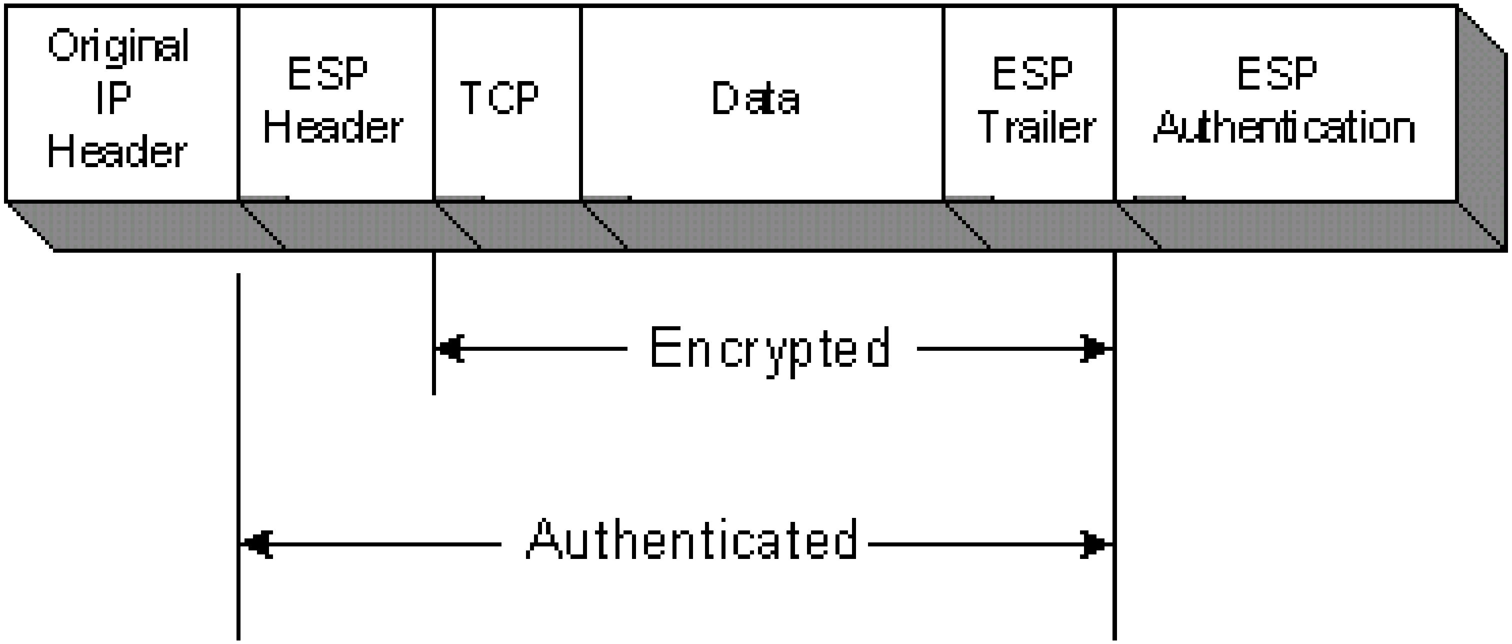
IPSEC

Authentication Header (AH)

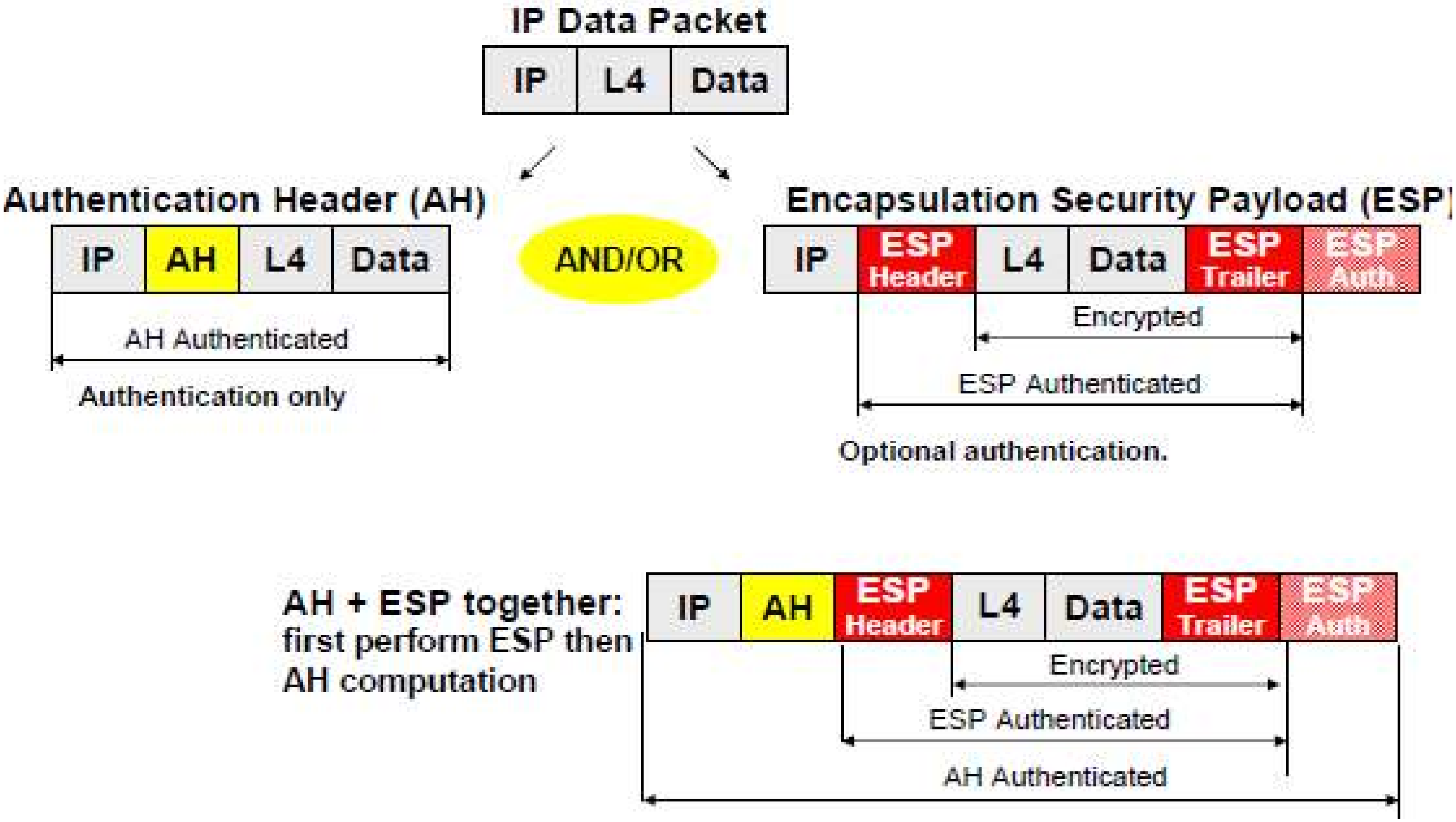


IPSEC

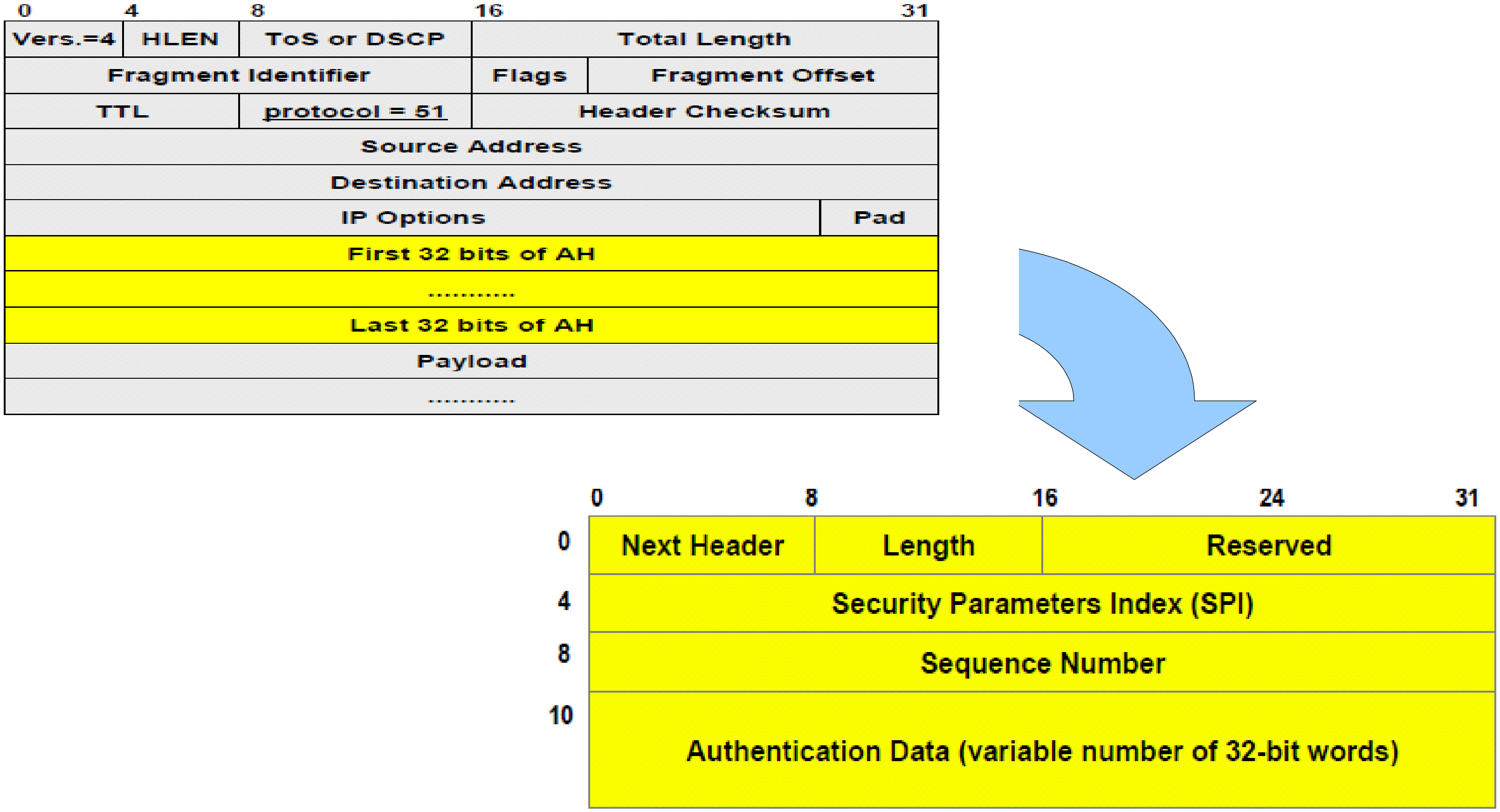
Encapsulating Payload Protocol (ESP)



IPSEC


IP Data Packet
IP
L4
Data
Authentication Header (AH)
IP
AH
L4
Data
AH Authenticated
Authentication only
AND/OR
Encapsulation Security Payload (ESP)
IP
ESP Header
L4
Data
ESP Trailer
ESP Auth
Encrypted
ESP Authenticated
Optional authentication.
AH + ESP together:
first perform ESP then
AH computation
IP
AH
ESP Header
L4
Data
ESP Trailer
ESP Auth
Encrypted
ESP Authenticated
AH Authenticated

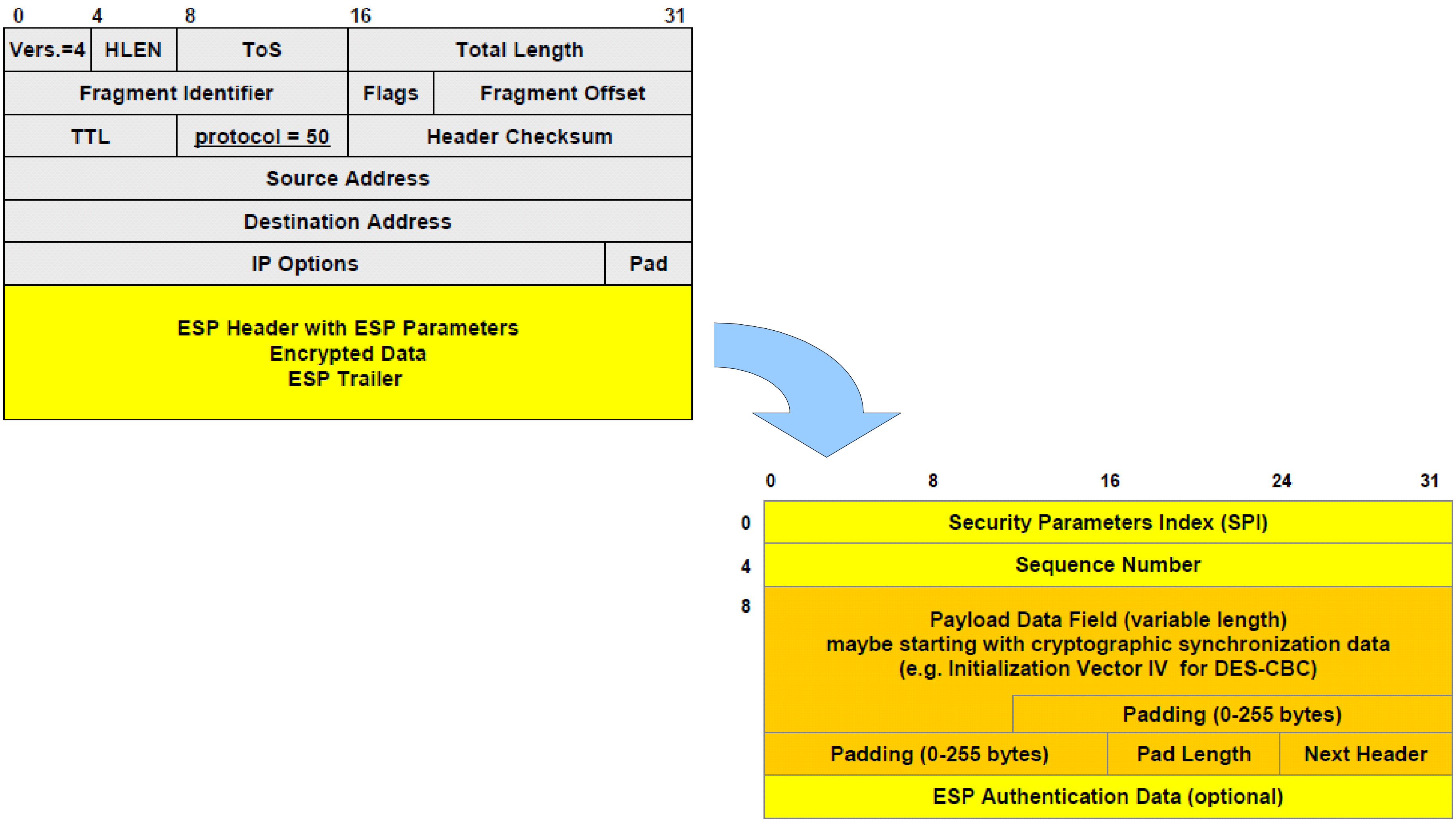
Authentication Mode



0	4	8	16		31
Vers.=4	HLEN	ToS or DSCP	Total Length		
Fragment Identifier			Flags	Fragment Offset	
TTL		protocol = 51	Header Checksum		
Source Address					
Destination Address					
IP Options					Pad
First 32 bits of AH					
…………					
Last 32 bits of AH					
Payload					
…………					

	0	8	16	24	31
0	Next Header	Length	Reserved		
4	Security Parameters Index (SPI)				
8	Sequence Number				
10	Authentication Data (variable number of 32-bit words)				

ESP



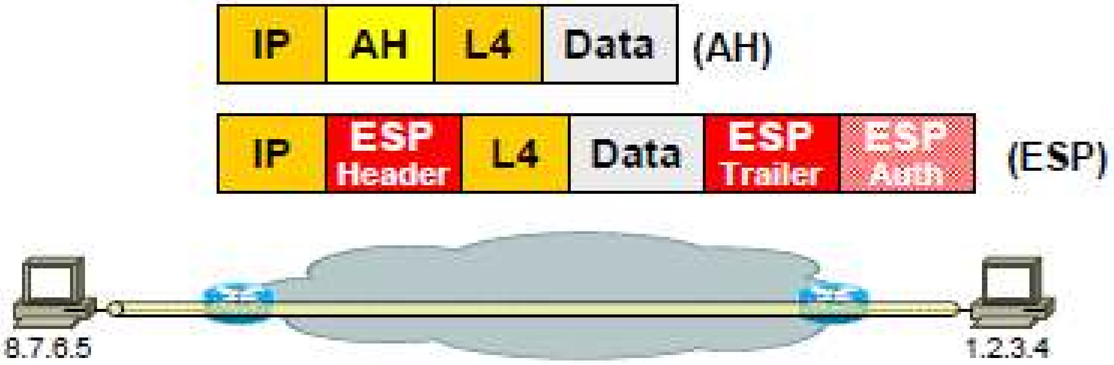
0	4	8	16	31
Vers.=4	HLEN	ToS	Total Length	
Fragment Identifier			Flags	Fragment Offset
TTL		protocol = 50	Header Checksum	
Source Address				
Destination Address				
IP Options				Pad
ESP Header with ESP Parameters Encrypted Data ESP Trailer				

	0	8	16	24	31
0	Security Parameters Index (SPI)				
4	Sequence Number				
8	Payload Data Field (variable length) maybe starting with cryptographic synchronization data (e.g. Initialization Vector IV for DES-CBC)				
		Padding (0-255 bytes)			
	Padding (0-255 bytes)		Pad Length	Next Header	
	ESP Authentication Data (optional)				

IPSEC

Transport Mode

- Only one IP header.
- Used for end-to-end sessions
- Does not hide communication statistics because of network header (IP addresses of the end systems) is sent in clear text

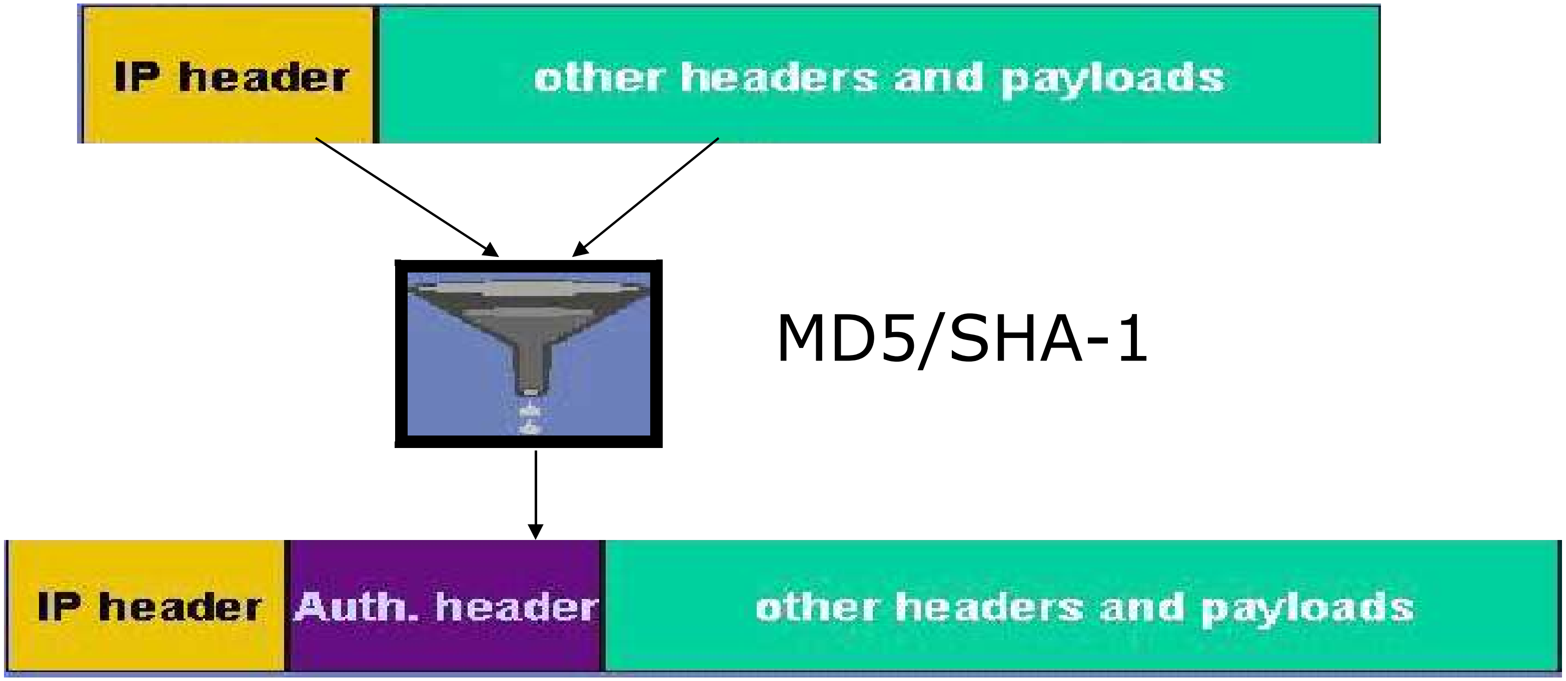


IP | AH | L4 | Data (AH)

IP | ESP Header | L4 | Data | ESP Trailer | ESP Auth (ESP)

IPSEC **Authentication Header** (AH)

Original IP packed

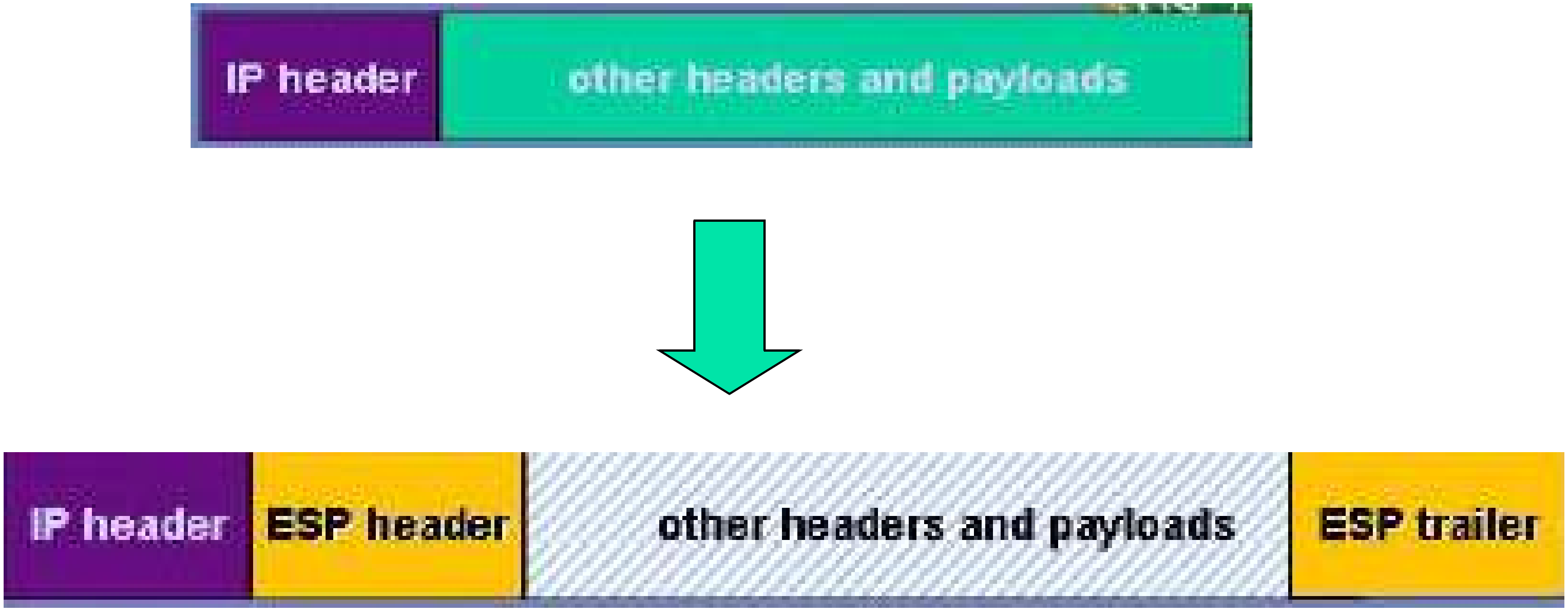


MD5/SHA-1

Authenticated packet

IPSEC: **ESP in Transport Mode**

Original IP packet

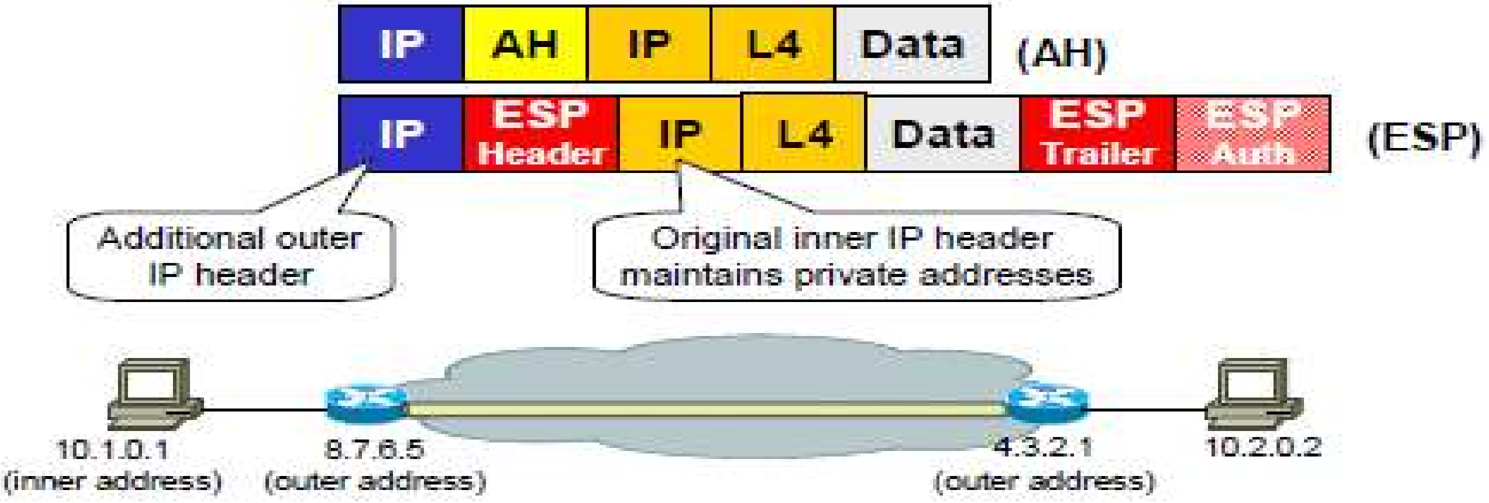


IP packet with ESP in Transport mode

IPSEC

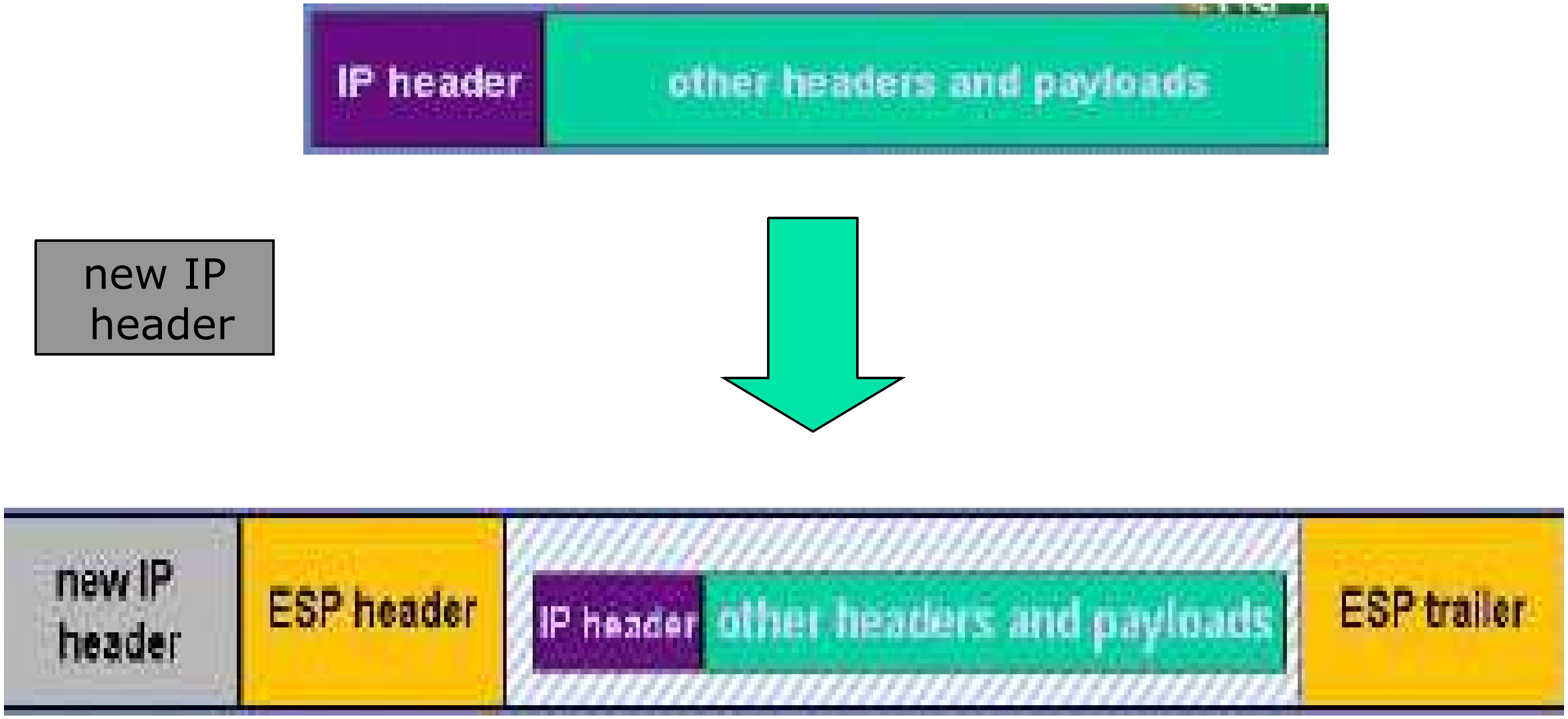
Tunnel Mode
for Site-to-Site VPN

- Whole original IP packet is IPsec-encapsulated.
- Used for VPNs.
- Does hide traffic patterns!


IP | AH | IP | L4 | Data
(AH)
IP | ESP Header | IP | L4 | Data | ESP Trailer | ESP Auth
(ESP)
Additional outer IP header
Original inner IP header maintains private addresses
10.1.0.1
(inner address)
8.7.6.5
(outer address)
4.3.2.1
(outer address)
10.2.0.2

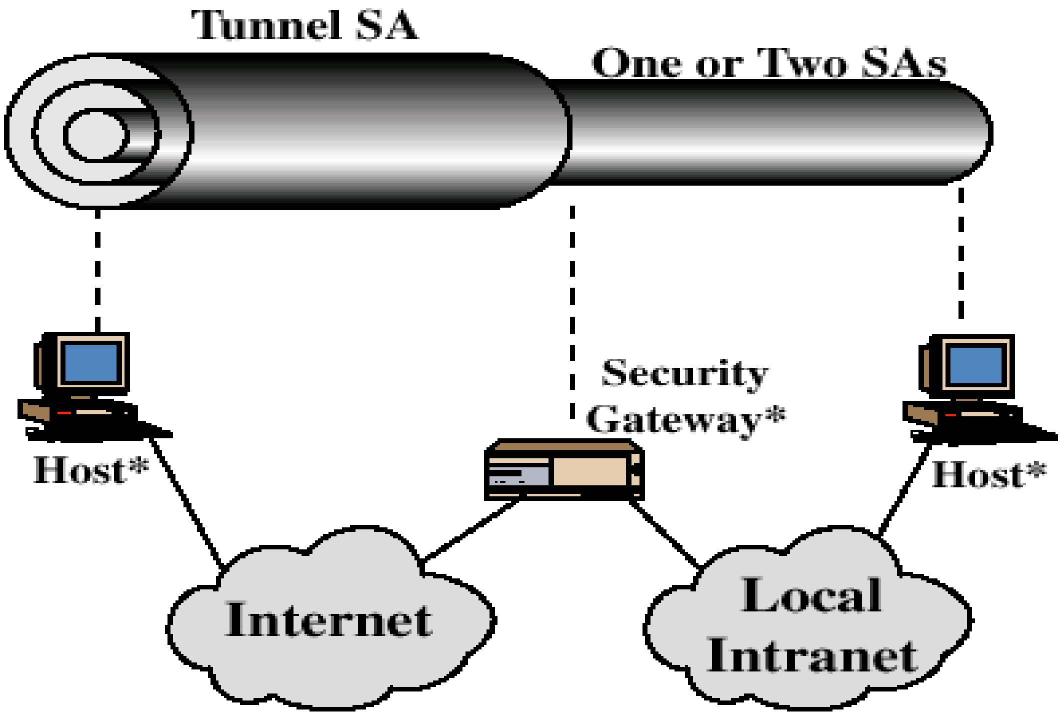
IPSEC: **ESP in Tunnel Mode**

Original IP packet



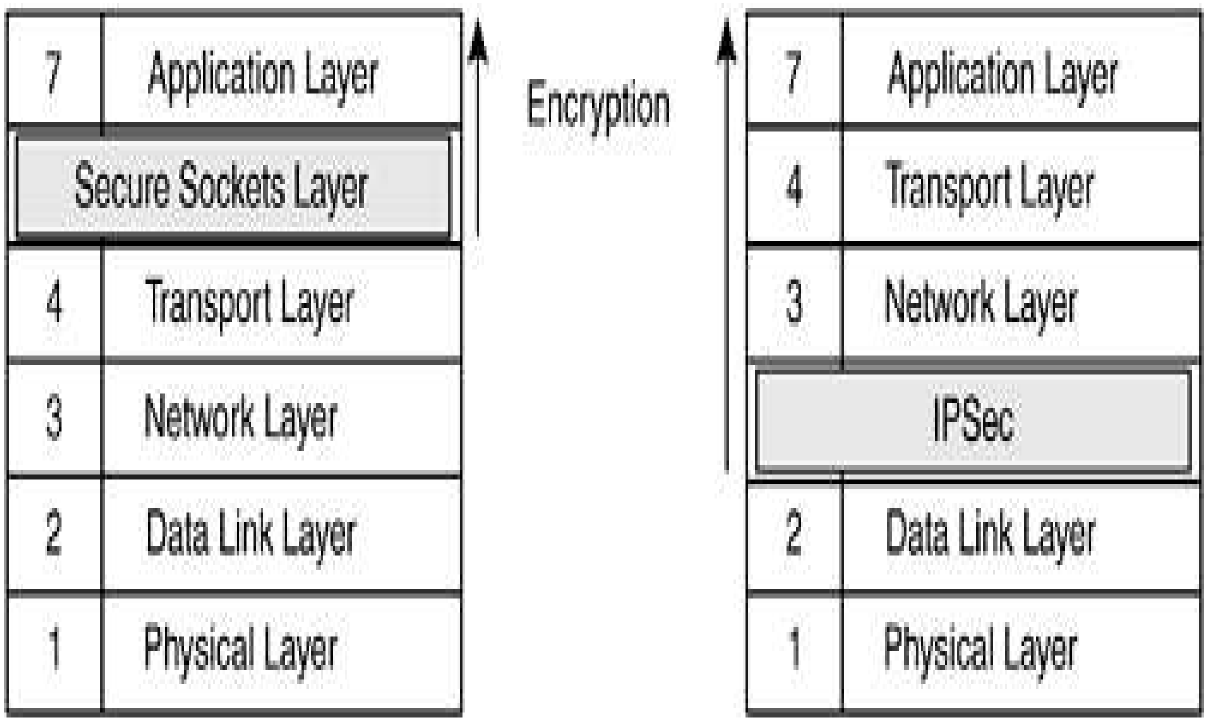
IP packet ESP + Tunnel mode

Applying several SAs



(d) Case 4

SSL vs IPSEC



7 Application Layer

Secure Sockets Layer

4 Transport Layer

3 Network Layer

2 Data Link Layer

1 Physical Layer

Encryption

7 Application Layer

4 Transport Layer

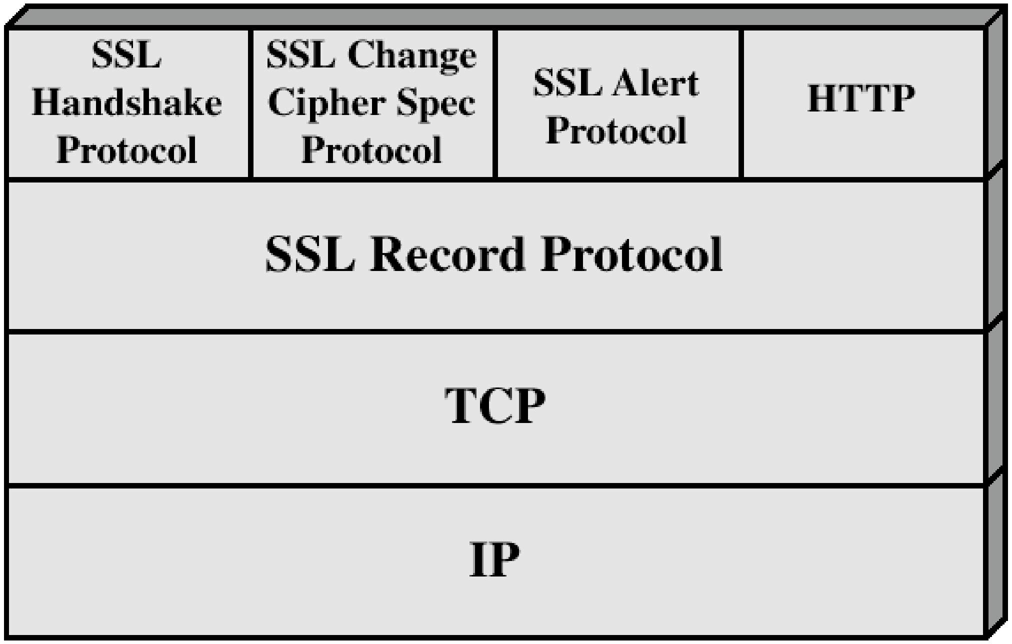
3 Network Layer

IPSec

2 Data Link Layer

1 Physical Layer

SSL = applicative VPN



SSL Handshake Protocol

SSL Change Cipher Spec Protocol

SSL Alert Protocol

HTTP

SSL Record Protocol

TCP

IP

Figure 7.2 SSL Protocol Stack

Four protocols

SSL

- Fragment, at most 16384 bytes (2**14)
- SSLv3 does not specify a compression method
  - No information loss, and length increase should be lower than 1024
  - Default = no compression
- Encryption methods
  - Idea (128) des (56) triple des (168)
  - Stream cipher: rc4-40, rc4-128

Some definitions

- session:
  - association between a client and a server that defines a set of parameters such as algorithms used, session number etc.
  - a session is created by the Handshake Protocol that allows parameters to be shared among the connections made between the server and the client. Sessions avoid negotiation of new parameters for each connection.
- connection: logical client/server link, associated with the provision of a suitable type of service. In SSL terms, it is a peer-to-peer connection with two network nodes.
- A single session is shared among multiple SSL connections between the client and the server. Multiple sessions may be shared by a single connection, but this is not used in practice.

Session state

Session identifier: an arbitrary byte sequence, chosen by the server to identify the state of an active session and can be reused to continue the session ;

- Peer certificate: the node certificate that may not exist;
- Compression method: the algorithm to compress the data;
- Cipher spec: the encryption algorithm and the one use to compute the MAC. It also defines cryptographic attributes as the hash_size;
- Master secret: a 48 byte secret information shared by the client and the server that will be used to compute the encryption keys;
- Is resumable: a flag that shows if the session can be reused

Connection State

The connection state is defined by the following parameters:

- Server and client random: a random byte sequence chosen by the client and by the server for each connection ;
- Server write MAC secret: to compute the MAC on the server data ;
- Client write MAC secret: to compute the MAC on the client data;
- Server write key: to encrypt the data server → client ;
- Client write key: to encrypt the data client → server ;
- Initialization vectors: a data for Cipher Block Chaining encryption. is shared by both partners because it is needed both to encrypt and decrypt.
- Sequence numbers 0.. 2^{64}-1: each partner stores and manages sequence numbers to send and receive messages on each connection. A number is zeroed when one partner sends a change cipher spec.

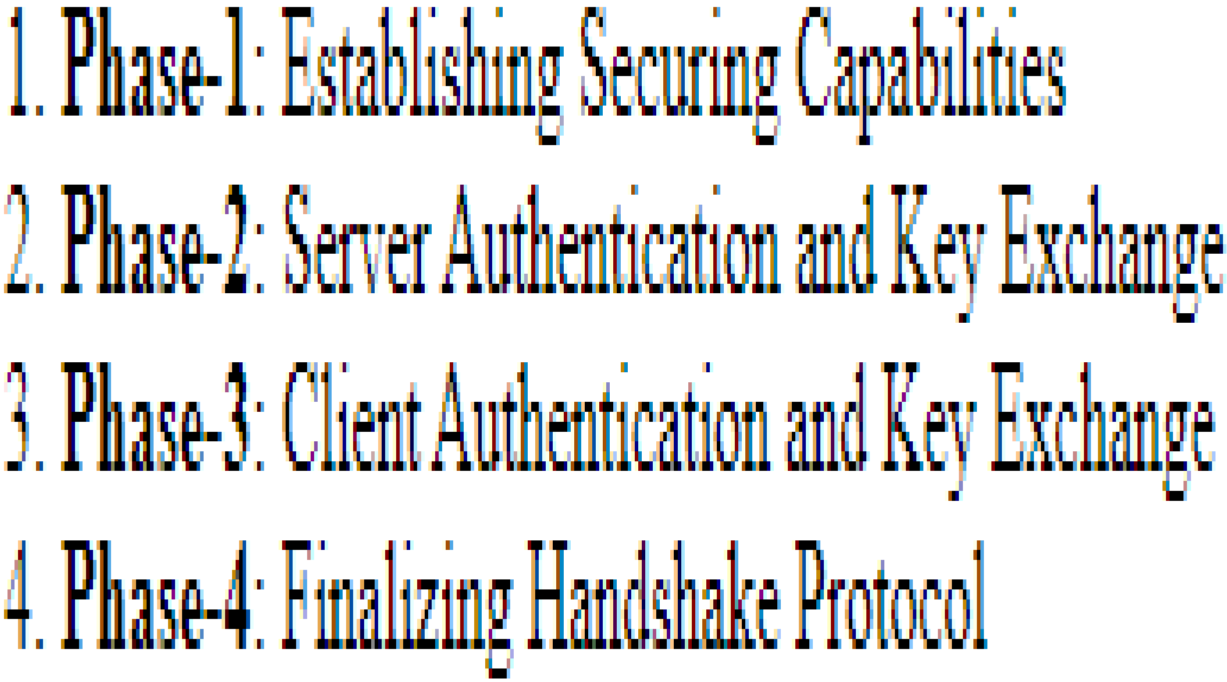
Record Protocol

- Frames and encrypts upper level data into one protocol for transport through TCP (reliable communications)
- 5 byte frame
  - 1st byte protocol indicator
  - 2nd byte is major version of SSL
  - 3rd byte is minor version of SSL
  - Last two bytes indicate length of data inside frame, up to 2^{14}
- Message Authentication Code (MAC)

The Four Protocols

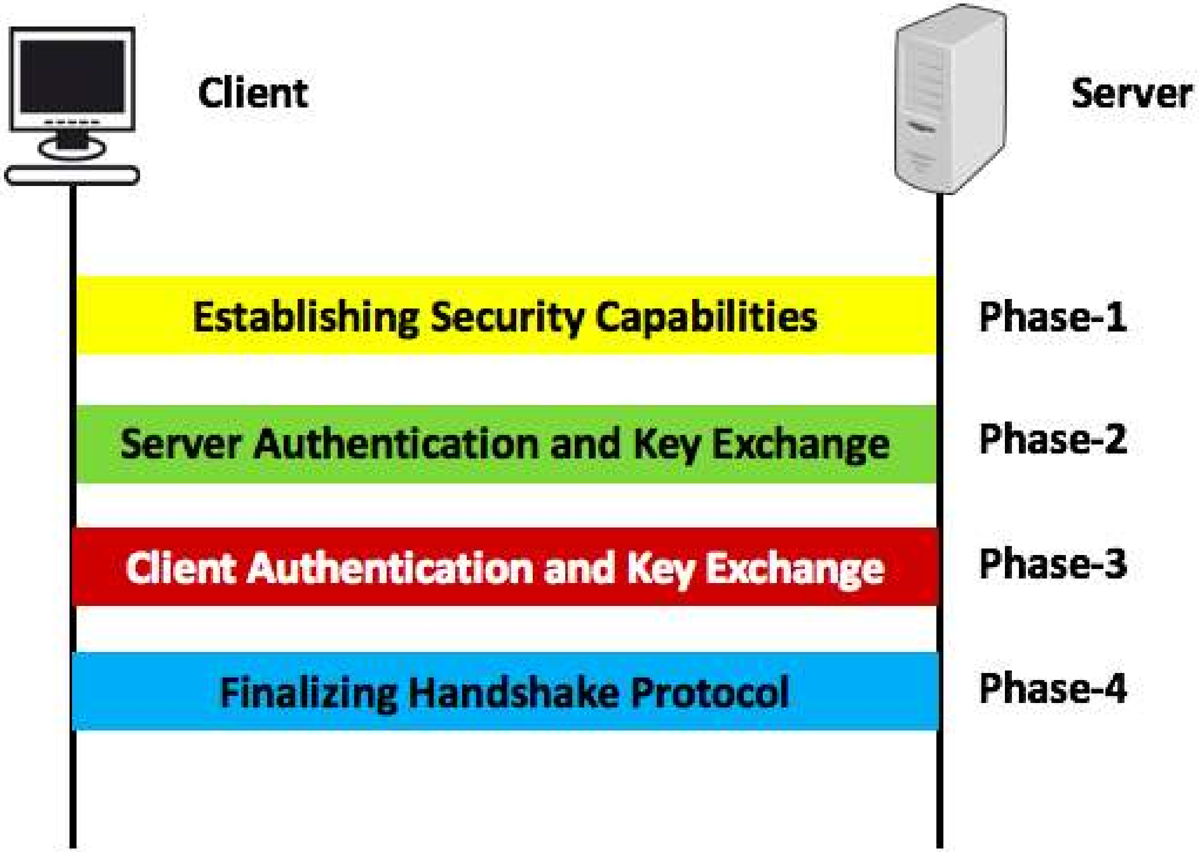
- Handshaking Protocol
  - Establish communication variables
- ChangeCipherSpec Protocol
  - Alert to a change in communication variables
- Alert Protocol
  - Messages important to SSL connections
- Application Protocol: the one that is encrypted

Four phases of SSL



1. Phase-1: Establishing Securing Capabilities
2. Phase-2: Server Authentication and Key Exchange
3. Phase-3: Client Authentication and Key Exchange
4. Phase-4: Finalizing Handshake Protocol

Four Phases



Client

Server

Establishing Security Capabilities

Phase-1

Server Authentication and Key Exchange

Phase-2

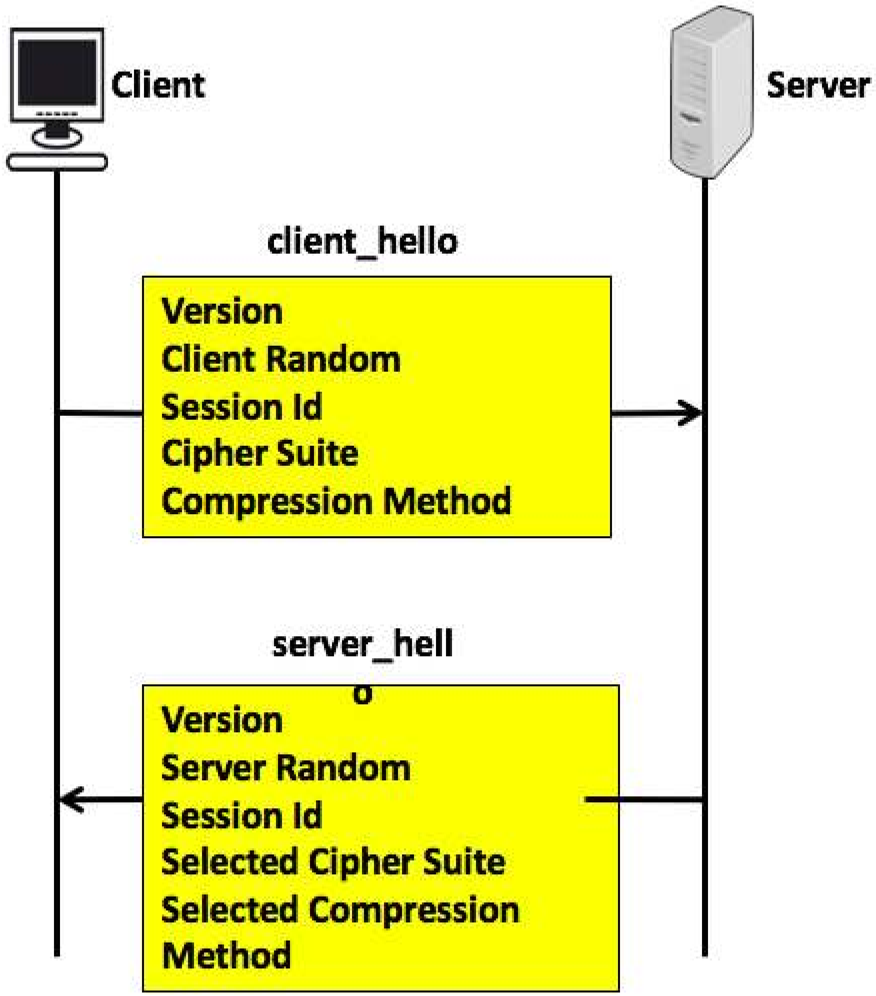
Client Authentication and Key Exchange

Phase-3

Finalizing Handshake Protocol

Phase-4

Establishing security capabilities



Establishing security capabilities

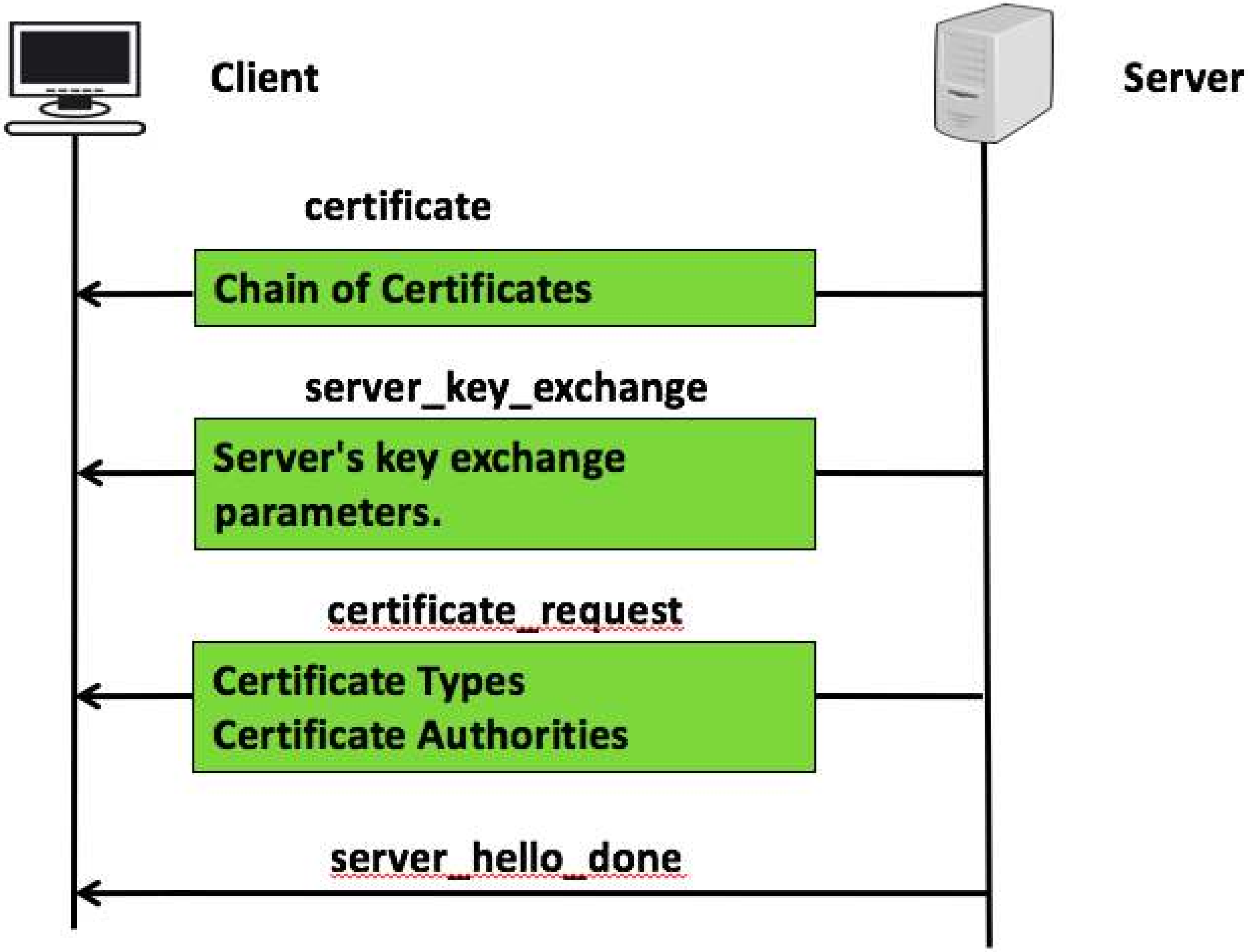
- It exchanges security capabilities, started by the client_hello message from the client to the server. It contains various parameters:
- client_hello:
  - version:
  - client random : 32 bits timestamp + 28 bytes of random generated by client
  - session Id: variable session length, 0 mean new session, else client want to update existing session.
  - cipher Suite : list of the course for decreasing order like keys, encryption methodology, etc..
  - compression Method: method which used for compression etc

Establishing security capabilities

server_hello

- version: either send by client or server version if
- server random: similar type of client session but independent of client.
- session Id: if client id is 0 server put new session id which indicates new session else client id
- selected Cipher Suite: selected suites by client
- selected compression method: selected compression algorithm used in during transfer.

Phase-2: Server Authentication and Key Exchange

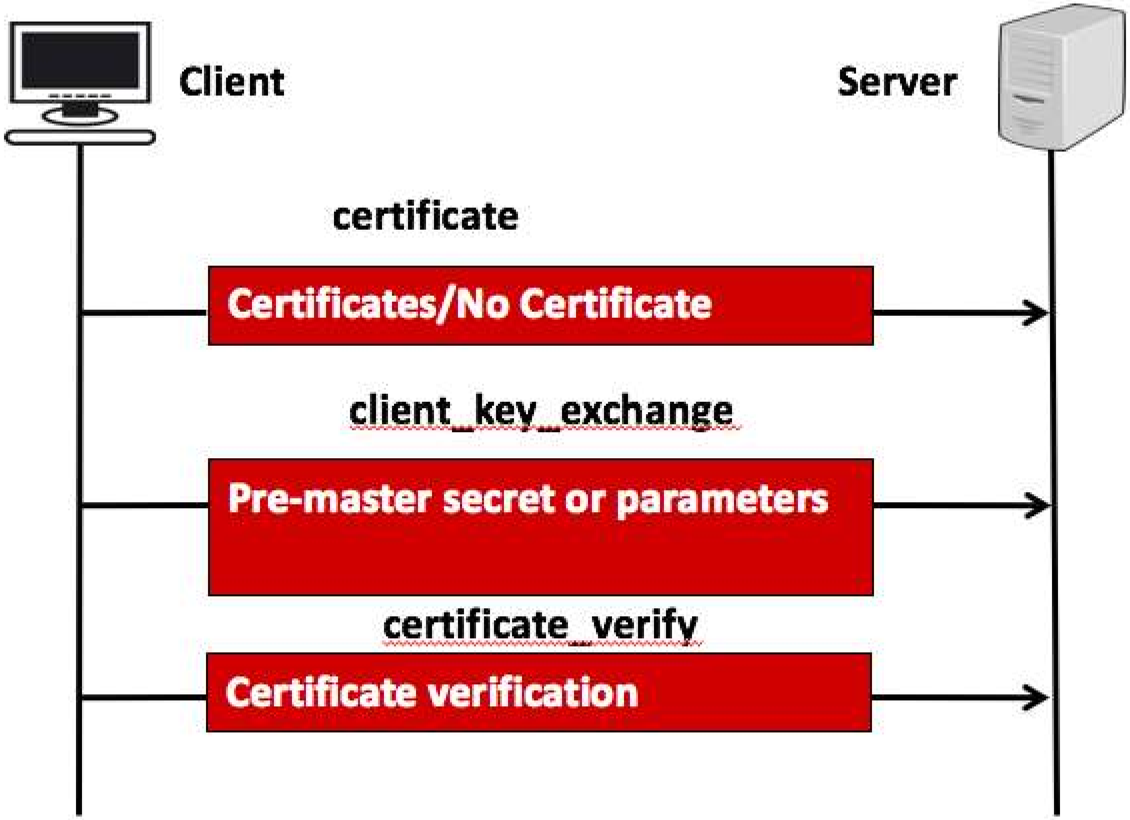


Phase-2: Server Authentication and Key Exchange

The server sends to the client.

- Certificate: it conveys server certificate to the client.
- server_key_exchange: server key exchange parameters sent if server certificate does not contains enough data to allow the client master pre secret.
- certificate_request: server request certificate from the client it has two things: certificates types with list of certificate types client may offer and list of distinguished name of acceptable certificate authorities.
- server_hello_done: no parameter, it indicates client can proceed with this phase exchange.

Phase-3: Client Authentication and Key Exchange

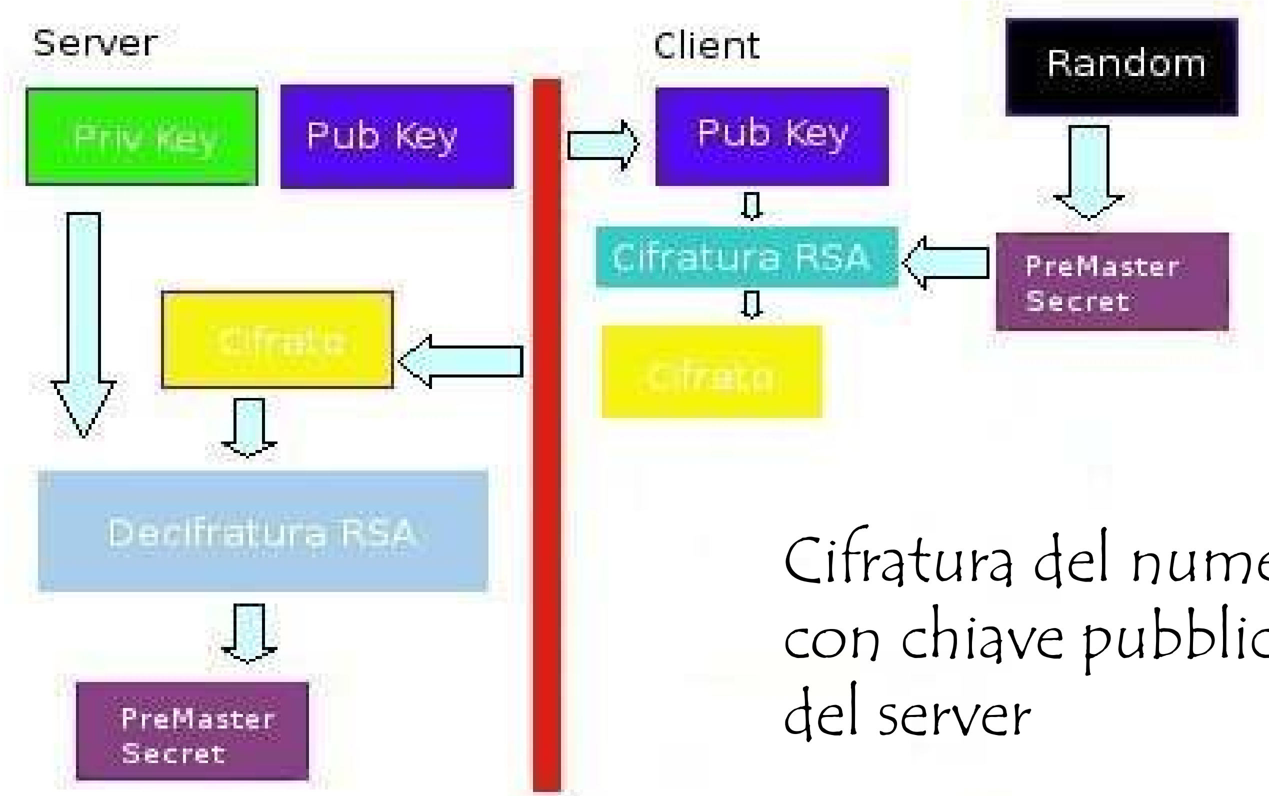


Phase-3: Client Authentication and Key Exchange

The client sends to the server.

- certificates: this message is sent if not suitable certificate is available, if some aspect change unexpectedly
- client_key_exchange: depending on the cipher suit selector on phase-1, with this message either pre master secret and parametrs sent which is used to calculate both side,
- certificate_verify: use to provide explicit verification of client certificate. it must immediately follows client_key_exchange.

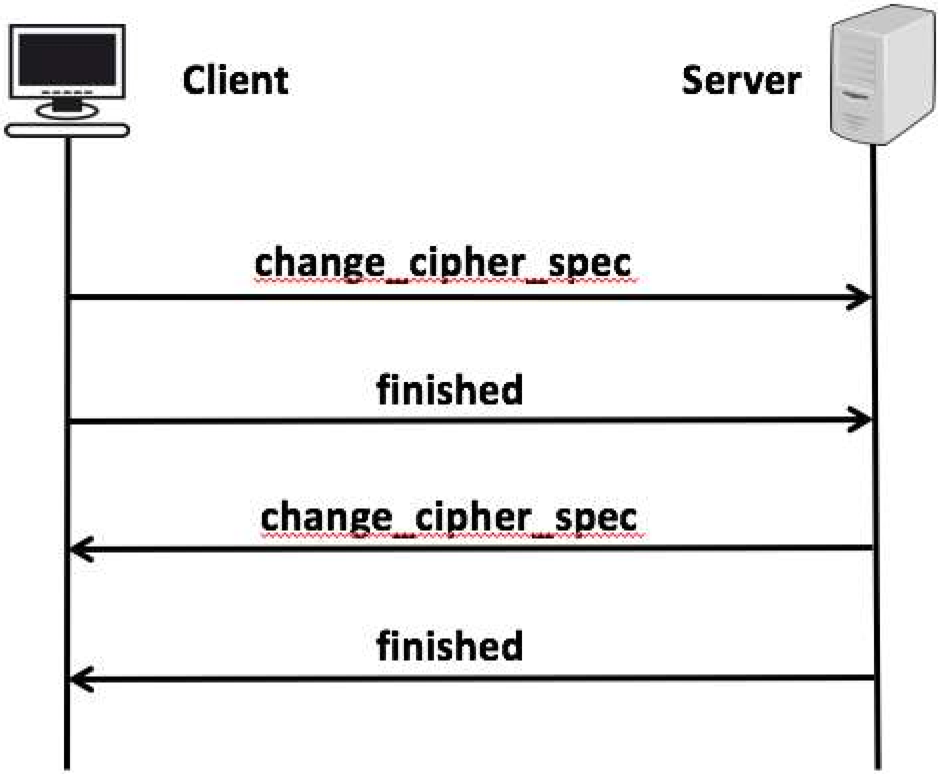
Premaster secret vs secret



Cifratura del numero random
con chiave pubblica
del server

master_secret = MD5(pre_master_secret || SHA(‘A’ || pre_master_secret || ClientHello.random || ServerHello.random) || MD5(pre_master_secret || SHA(‘BB’ || pre_master_secret || ClientHello.random || ServerHello.random)) || MD5(pre_master_secret || SHA(‘CCC’ || pre_master_secret || ClientHello.random || ServerHello.random));

Phase-4: Finalizing Handshake Protocol



Phase-4: Finalizing Handshake Protocol

- Once session has been established then SSL record protocol start sending data
- It is only session which enables multiple connections.
- The SSL protocol creates a session identified by cryptographic parameters. A session parameters can be used across multiple connections to avoid time consuming new security parameters.

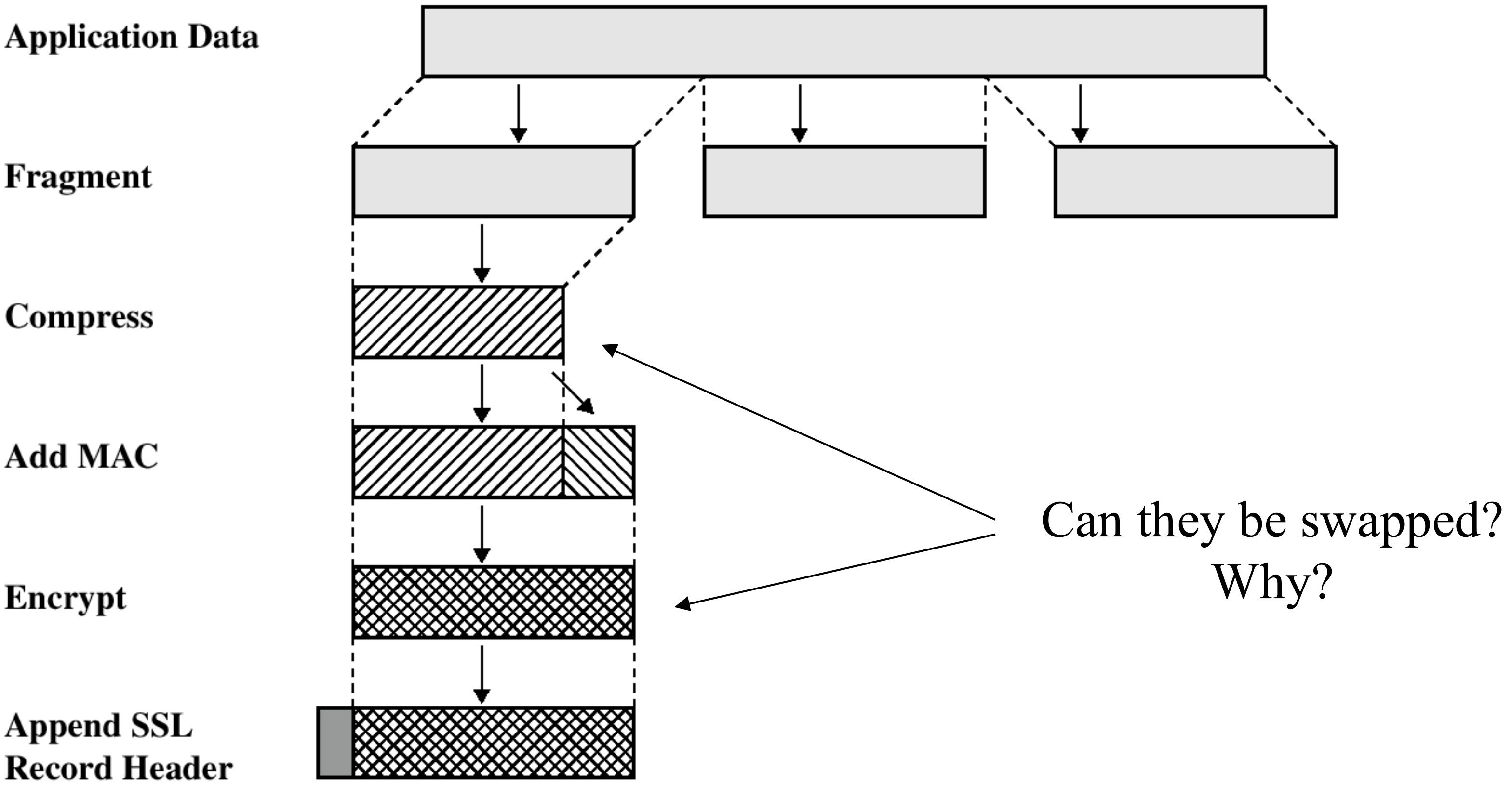
Detail ChangeCipherSpec Protocol

- Special protocol with only one message
- When Client processes encryption information, it sends ChangeCipherSpec message
  - Signals all following messages will be encrypted
- ChangeCipherSpec is always followed by Finished message

Detail: The End of the Beginning

- Upon receipt of ChangeCipherSpec, Server sends its own ChangeCipherSpec and Finished messages
- After both Client and Server receive Finish messages, Handshaking phase is over
- Any following communication is encrypted
- Encryption and compression methods can be changed with new ChangeCipherSpec messages

SSL – Record Protocol



Application Data

Fragment

Compress

Add MAC

Encrypt

Append SSL
Record Header

Can they be swapped?
Why?

Message Authentication Code

- MAC secures connection in two ways
  - Ensure Client and Server are using same encryption and compression methods
  - Ensure messages sent were received without error or interference
- Both sides compute MACs to match them
- No match = error or attack

MAC

hash(MAC_write_secret || pad_2 || hash(MAC_write_secret || pad_1 || seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

where :

- ||= concatenation;
- MAC_write_secret: secret shared key;
- hash: hash algorithm (MD5 o SHA-1);
- pad_1: byte 0x36 (00110110) repeated 48 times (384 bit) for MD5 and 40 (320 bit) for SHA-1;
- pad_2: byte 0x5C (01011100) repeated 48 times for MD5 and 40 for SHA-1;
- seq_num: sequential number of the message;
- SSLCompressed.type: higher level protocol to be applied;
- SSLCompressed.length: length of the compressed packet;
- SSLCompressed.fragment: compressed fragment (the clear text fragment if no compression is applied).

Alert and Application Protocols

- Alert protocol always two byte message
  - First byte indicates severity of message
    - Warning or Fatal
    - A Fatal alert will terminate the connection
  - Second byte indicate preset error code
  - Secure connection end alert not always used
- Application Protocol is HTTP, POP3, SMTP, or whatever application is being used
  - Simply give a datagram to the Record Layer

Alert = Exception

- unexpected_message;
- bad_record_mac;
- decompression_failure;
- handshake_failure: the sender cannot negotiate an acceptable set of parameters
- illegal_parameter: an uncorrect handshake parameter.
- close_notify: sent by each side before closing its side of the connection
- no_certificate: reply if no certificate can be used ;
- bad_certificate: the received certificate has been manipulated
- unsupported_certificate: the receiver certificate is not supported ;
- certificate_revoked, _expired, _unknown: the certificate has been revoked, or is out of date or it cannot be elaborated

Benefits

- Ease of implementation
  - For network application developers
    - As easy as implementing unsecured Sockets
  - For network implementation developers
    - Simply add layer to established network protocol stack
  - For Users
    - Only need to authorize certificates

Drawbacks

- More bandwidth needed
- Slower
- Needs a dedicated port – 443 for HTTPS
- Assumes reliable transport for underlying transport protocol
  - No UDP
  - Implications for streaming media, VoIP

HeartBleed

- A vulnerability discovered in April of 2014; it allowed attackers unprecedented access to sensitive information, and it was present on 17% web servers,
- Caused by a flaw in OpenSSL, an open source code library that implemented the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols.
- A malicious user could easily trick a vulnerable web server into sending sensitive information, including usernames and passwords.

HeartBleed

- A heartbeat is an important part of TLS/SSL protocols
- Essentially, it is how two communicating computers let each other know they are still connected even if the user isn't downloading or uploading anything at the moment.
- Occasionally, one of the computers will send an encrypted piece of data, a heartbeat request, to the other. The second computer will reply back with the exact same encrypted piece of data, proving that the connection is still in place. Crucially, the heartbeat request includes information about its own length.

HeartBleed

- If you haven't done anything in a while your browser might send a signal to webmail servers saying

  "This is a 40 KB message you wiil get. Repeat it all back to me."

- When servers receive that message,
  - they allocate a 40 KB memory buffer based on the reported length of the heartbeat request.
  - they store the encrypted data from the request into that memory buffer,
  - They read the data back out of it and sends it back to the browser.

HeartBleed Vulnerability

- OpenSSL implementation of the heartbeat functionality was missing a crucial safeguard: the server that received the request never checked to make sure the request was actually as long as it claimed to be.
- So if a request claims it was 40 KB long but was actually only 20 KB, the receiving server would set aside 40 KB of memory buffer, then store the 20 KB it actually received, then send back that 20 KB plus whatever happened to be in the next 20 KB of memory.
- That extra 20 KB of data is information that the attacker has now extracted from the web server.

HeartBleed Attack

- The attacker has no way to know in advance what might be lurking in that 20 KB grabbed off the server
- There are a number of possibilities. It could be
  - gibberish or useless cruft
  - SSL private keys, which would enable the decryption of secure communication to that server unlikely, but the holy grail for an attacker).
  - usernames and passwords that had been submitted to applications and services running on the server.

HeartBleed Code

- It's not clear if any real-world exploitation of the vulnerability took place before it was widely publicized.
- Attempted attacks detected by security companies as early as 2013 maybe were probing the vulnerability and some think the attackers were government agencies.
- After April of 2014, when the vulnerability was made public, companies scrambled to update their systems, but several attacks where successful
- An attack on Community Health Systems that stole patient data was blamed on Heartbleed, as the theft of hundreds of social ID numbers from the Canadian Revenue Agency.

HeartBleed Attacker

- The coding mistake that caused Heartbleed is due to a single line of code:

```
memcpy(bp, pl, payload)
```

- The command copies data to bp from pl and payload is the length of the data being copied.
- The problem is that there's never any attempt to check if the amount of data in pl is equal to the value given of payload.

VPN – The unsolved problem

- If a node hosts some malware, connecting it through a vpn does not solve the problem
- The trust on a machine strongly depends upon the user and the machine status
- Encryption neglects the context and hence cannot solve all the problems
- We need distinct inventories

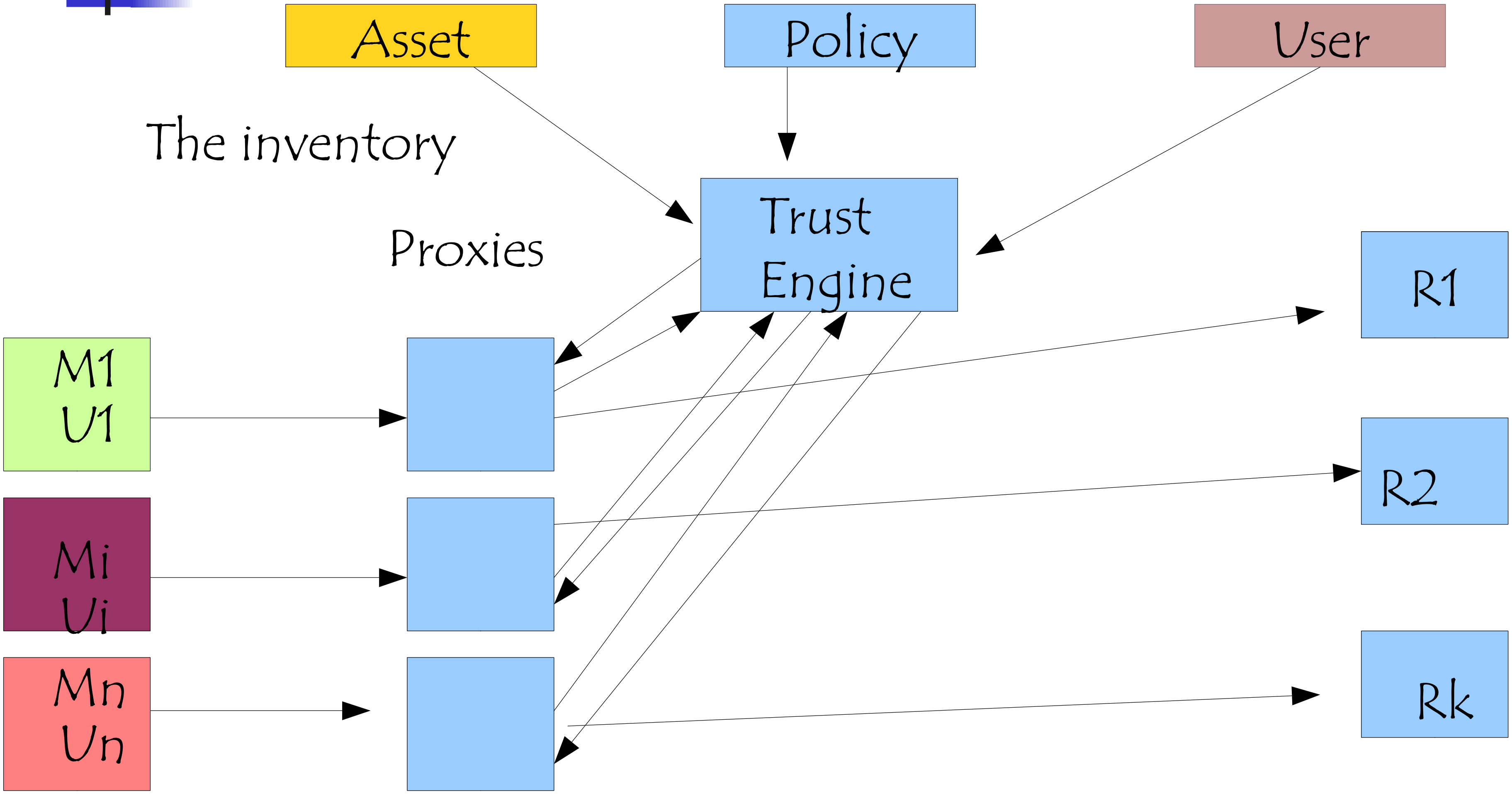
Inventories

- Asset inventory:
  - for each machine information on the os, the ids, the patches that have been applied
  - for some machines no info
- User inventory:
  - the role of each user,
  - the resources it needs to access
- Certificates to prove the identity of a machine and of a user

Trust engine

- When a user A wants to access a resource R using a machine M, the engine computes a trust value
- How much we can trust A using M to access R
- Access is granted according to the security policy that pairs each R with a trust threshold

Zero Trust – The future of VPN



Proxies

- Google is proposing a system design approach that strongly uses proxy to decouple from
  - Security
  - Reliability

  the various components of an application
- Properties of a proxy can be implemented and tested more easily due to the low amount of software
- Proxies can also simplify load balancing and handle request overloads

Zero Trust

- The proxy receives a request
- It transmits the request parameters and the certificates to the trust engine
- If the engine grant access, the proxy
  - creates an encrypted connection to the machine
  - interfaces only the access to a resource or even the requests
  - can also act as a load balancer but its main role is to enforce control
- Proxy = firewall for services

Proxy Advantages

- User Identity Awareness and Protectio.
- Service Usage, Visibility and Control:
- Secure Encrypted Traffic
- Detect and Prevent Advanced Threats by content analysis and anomaly detection
- Functions scalability
- Deployment Flexibility: proxies for distinct services with distinct security requirements

Zero Trust

- Identities of the machine and of the user are verified using certificates
- Problems arise to store the certificates in a safe and secure way
- In particular, we have to relate the configuration of the machine and the identity
- Identity is not a serial number, but a serial number, an OS, an OS configuration ….

Zero Trust

- It overcomes perimeter defence us vs them), the access is granted according to the status of the requester (user+machine) and not according its physical position or proximity to the resource
- Generalizes defence -in-depth
- Critical point: asset inventory, zero trust cannot be adopted if the inventory cannot be built
- This points out that no security without inventories
- The least privilege strikes back :-D

Countermeasures - OS

- An OS that can offer a large set of security policy native rather than a predefined one
- Implemented by the OS, built in the OS, rather than emulated on top of the OS using the ones the OS defines
- Large set of choices = MAC + DAC + RBAC ...
- It increases the security of the applications of networking and of all the applications it supports

Security Enhanced Linux

- A set of mechanisms to implement MAC + DAC security policies as OS policy
- A set of tools that support
  - A simple description of the security policy of interest
  - Check the consistency of the description
  - Produce the information to enable the adoption of the policy by the OS
- Evolution of two OSs: Flask e Fluke
- Both are microkernel OS
- NSA + NAI + MITRE

SELinux – NSA statement

The increased awareness of the need for security has resulted in an increase of efforts to add security to computing environments. However, these efforts suffer from the flawed assumption that security can adequately be provided in application space without certain security features in the operating system. In reality, operating system security mechanisms play a critical role in supporting security at higher levels. This has been well understood for at least twenty five years and continues to be reaffirmed in the literature. Yet today, debate in the research community as to what role operating systems should play in secure systems persists. The computer industry has not accepted the critical role of the operating system to security, as evidenced by the inadequacies of the basic protection mechanisms provided by current mainstream operating systems. The necessity of operating system security to overall system security is undeniable; the operating system is responsible for protecting application-space mechanisms against tampering, bypassing, and spoofing attacks. If it fails to meet this responsibility, system-wide vulnerabilities will result.

SELinux – NSA: an update

The increased awareness of the need for security has resulted in an increase of efforts to add security to computing environments. However, these efforts suffer from the flawed assumption that security can adequately be provided in application space without certain security features in the operating system. However is simpler and resulting in a larger return for the computer industry to not accept the critical role of the operating system to security, as evidenced by the inadequacies of the basic protection mechanisms provided by current mainstream operating systems. Instead, the computer industry will offer further products to increase the very low security that a standard OS offers.

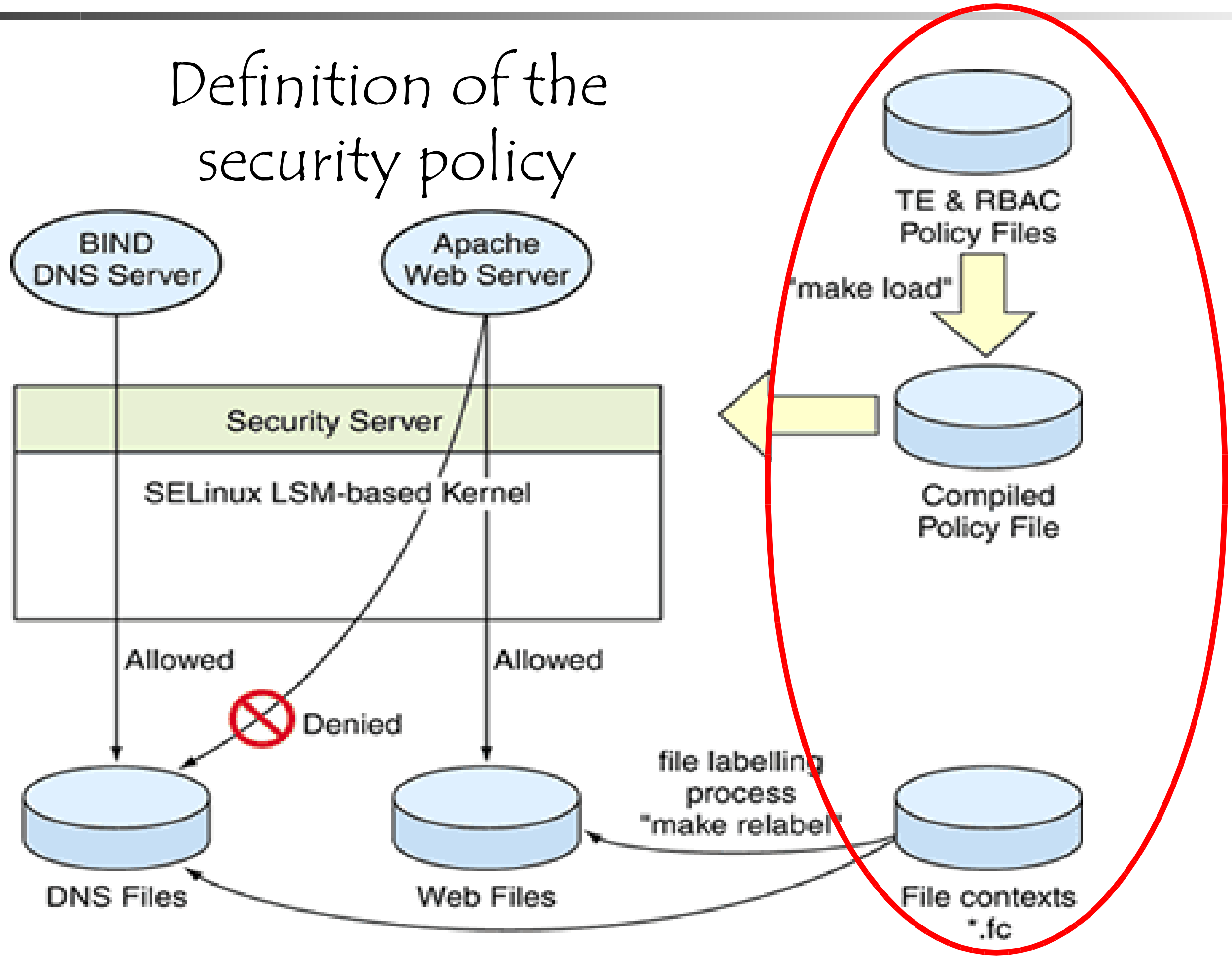
An interesting comment...

Let me assure you that this action by the NSA was the crypto-equivalent of the Pope coming down off the balcony in Rome, working the crowd with a few loaves of bread and some fish, and then inviting everyone to come over to his place to watch the soccer game and have a few beers.

There are some things that one just never expects to see, and the NSA handing out source code along with details of the security mechanism behind it was right up there on that list.

Why do we need a SE Linux rather than Linux?

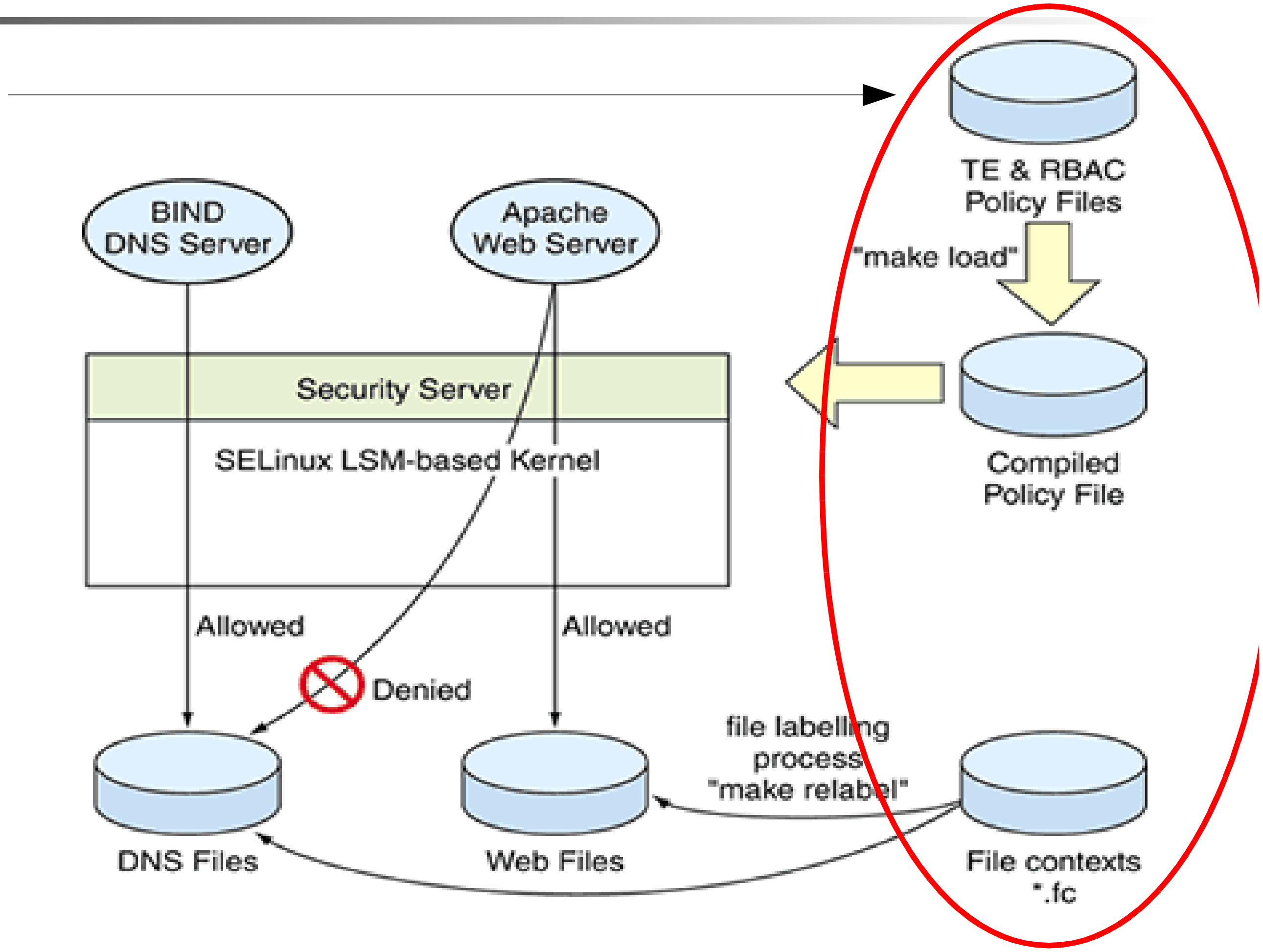
Definition of the security policy



Why do we need a SE Linux and not only Linux?

Definition of the security policy and its implementation

The least privilege principle strike again



SeLinux vs Linux

- Linux defines the user rights on resources
- Selinux defines
  - The rights of each program on resources
  - The programs that each user can run
- Rights are defined in terms of types, of roles and of levels
  - Type1 can do this op on type2
  - This role can run program with these types
  - level comparison (orthogonal)

SE - Linux

- Final goal: the security policy is a configuration parameter
- Both MAC and DAC security policy can be defined
- No notion of root user
- Model to define security policies is based upon two prototypes: Flask and Fluke

In brief – what SE Linux covers

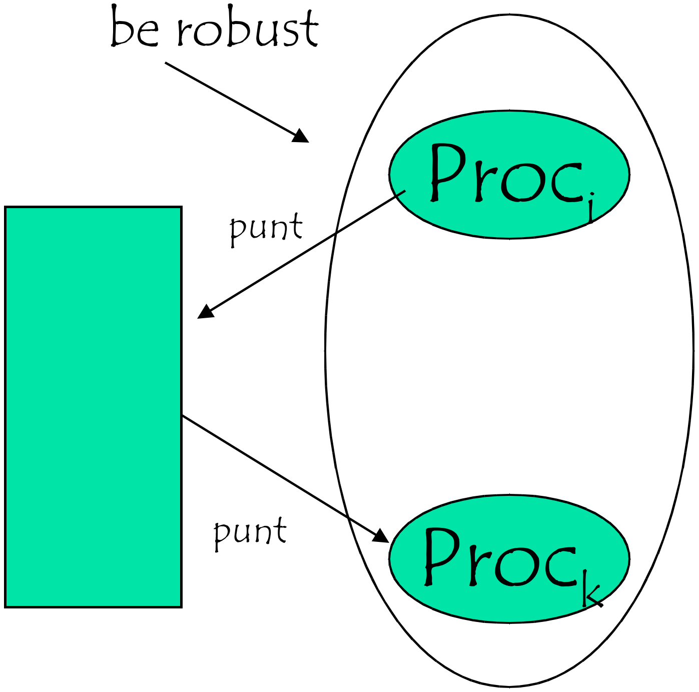
- DAC = Discretionary Access Control = user rights are defined by the owner
- MAC = Mandatory Access Control = system wide constrains that the owner has to respect
- RBAC = Role Based Access control = rigths defined according to the user role
- Role= set of users = distinct rights of the same user at distinct times
- MLS = multilevel security = MAC constrain defined in terms of levels of subjects and objects

General Model - SID

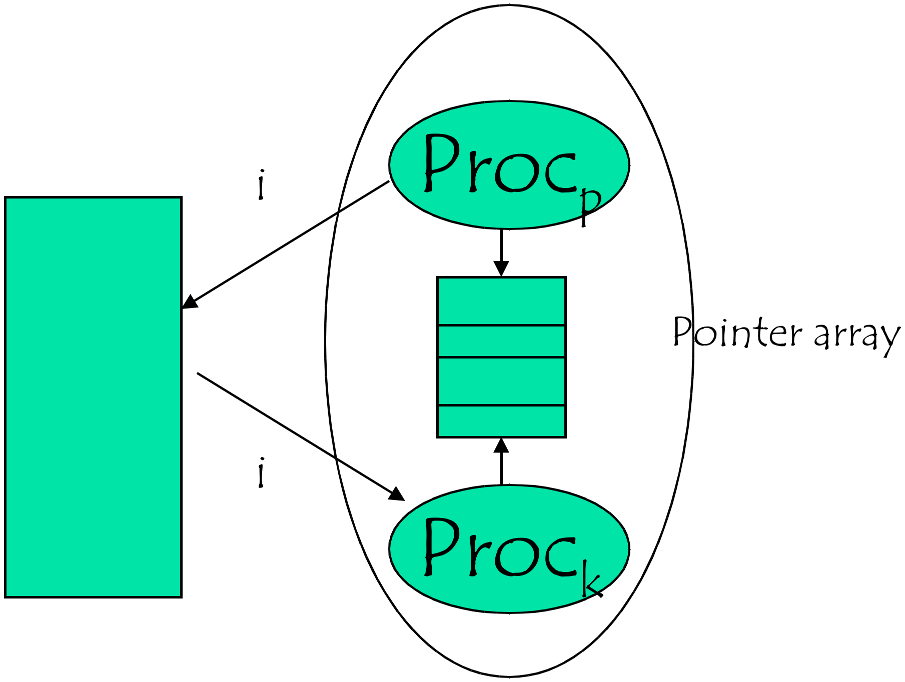
- Each subject and each object is paired with a security context, the one used to solve access control decisions
- Together with the user it defines the context = user, type, role, level
- This information is stored in a security server that is invoked before executing an operation
- Each process can only access a logical pointer to this context that it transmits to the server

We have already seen this Pointer - I

Package that should be robust

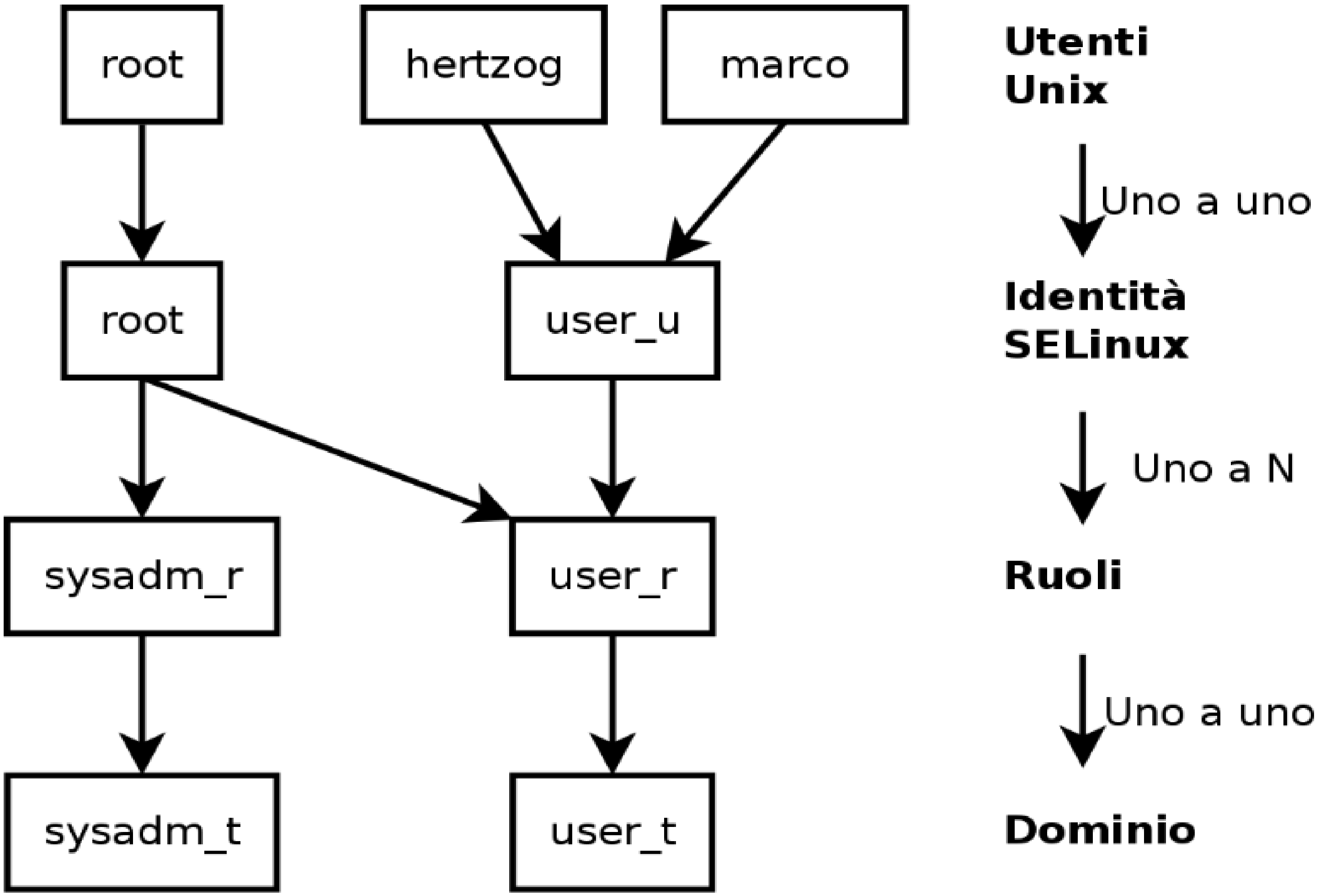


A more robust version



An index is transformed into a pointer by accessing the pointer array

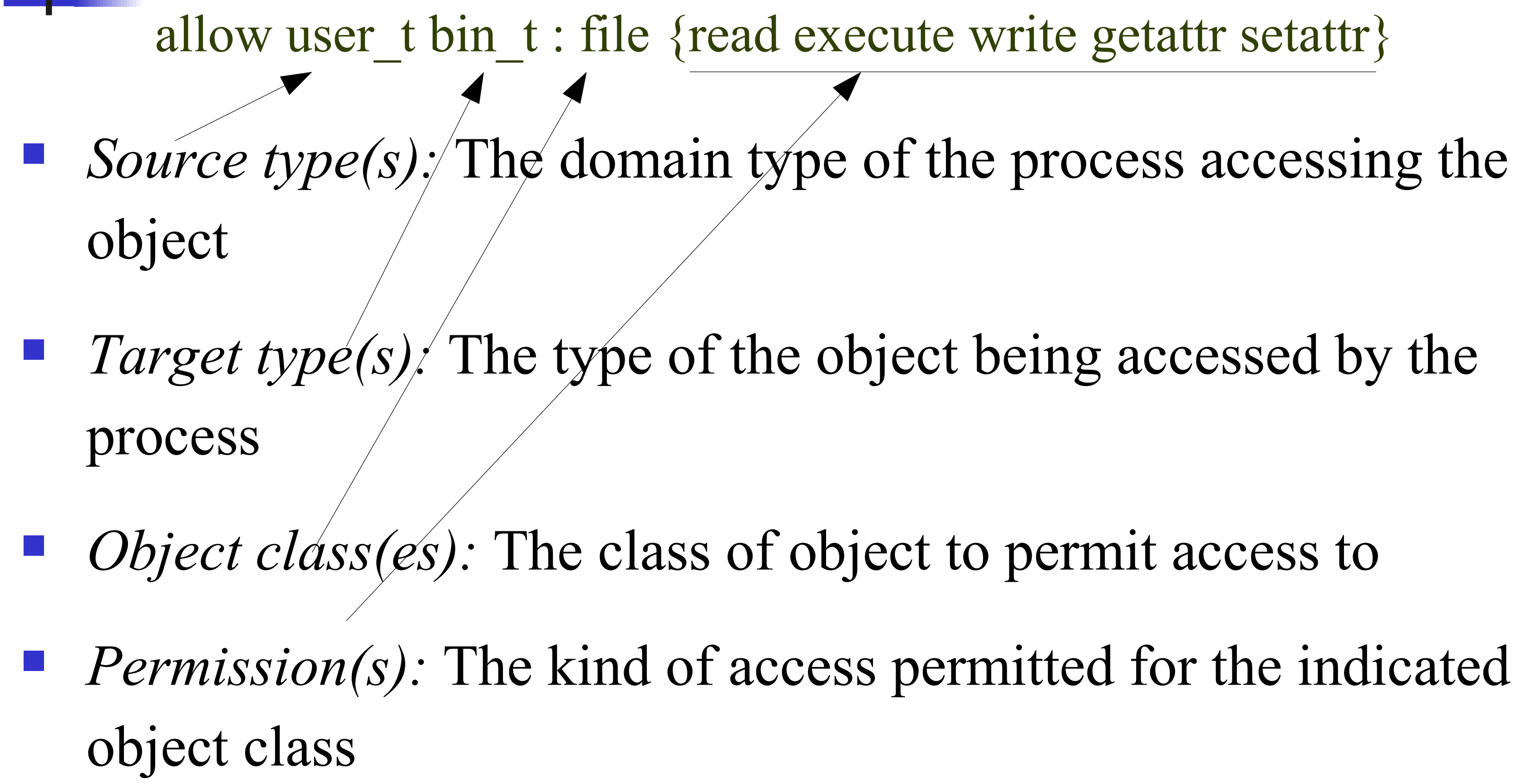
Relation among names, roles etc.



Type Enforcement

- *Object:* system item that is acted upon (files, IPC, sockets, etc….)
- *Subject:* process that is requesting access to an object
- All Objects and Subjects contain a security context
- *Security Context(s)* are composed of four parts:user, role, type, and level. Sometimes level is missing
- All Security Context components are checked against the policy to see if access is allowed.
- Type is the base component while role and user are used to further restrict type enforcement

TE Access Control



```
allow user_t bin_t : file {read execute write getattr setattr}
```

- *Source type(s):* The domain type of the process accessing the object
- *Target type(s):* The type of the object being accessed by the process
- *Object class(es):* The class of object to permit access to
- *Permission(s):* The kind of access permitted for the indicated object class

Type Enforcement

- Several major keywords
  - `type`
  - `attribute`
  - `typeattribute`
  - `typealias`
  - `allow`
  - `dontaudit`
  - `auditallow`
  - `neverallow`

Type Enforcement

```
rule_name src_type_set target_type_set : class_set perm_set;
allow user_t bin_t : file { read getattr } ;
allow user_t bin_t : dir { read getattr search } ;

#invalid since file does not have a search permission
allow user_t bin_t { file dir } {read getattr search } ;
#valid

#dontaudit when this access is denied
dontaudit httpd_t etc_t : dir search ;

#audit when this access is allowed
#by default allowed access is not audited
auditallow domain shadow_t : file write ;

#This statement may never be allowed by any rule
neverallow user_t shadow_t : file write

allow user_t bin_t : { file dir } * ;
allow user_t bin_t : file ~{ write setattr ioctl };
```

Domain Transitions

- Analogous to SetUID programs
- Joe running as user_t (untrusted user) needs to change his password. How does Joe change his password?
- allow user_t passwd_exec_t : file {getattr execute}
- allow passwd_t passwd_exec_t : file entrypoint
- allow user_t passwd_t : process transition
- Restricts trusted domain passwd_t and allows user_t to transition to it.
- Implicit domain transitions provided via type_transition.

Domain Transitions

- allow user_t passwd_exec_t : file {getattr execute}
- allow passwd_t passwd_exec_t : file entrypoint
- allow user_t passwd_t : process transition

=

- user_t can eexcute a file passwd_exec_t
- the execution of the file result in a role transition
- the transition between two roles is legal

Users & Roles

- Two components of a security context
- SELinux usernames and DAC usernames differs
- SELinux usernames are granted roles in the system
- Roles
  - collections of types geared towards a purpose
  - can be used to further restrict actions on the system

MLS

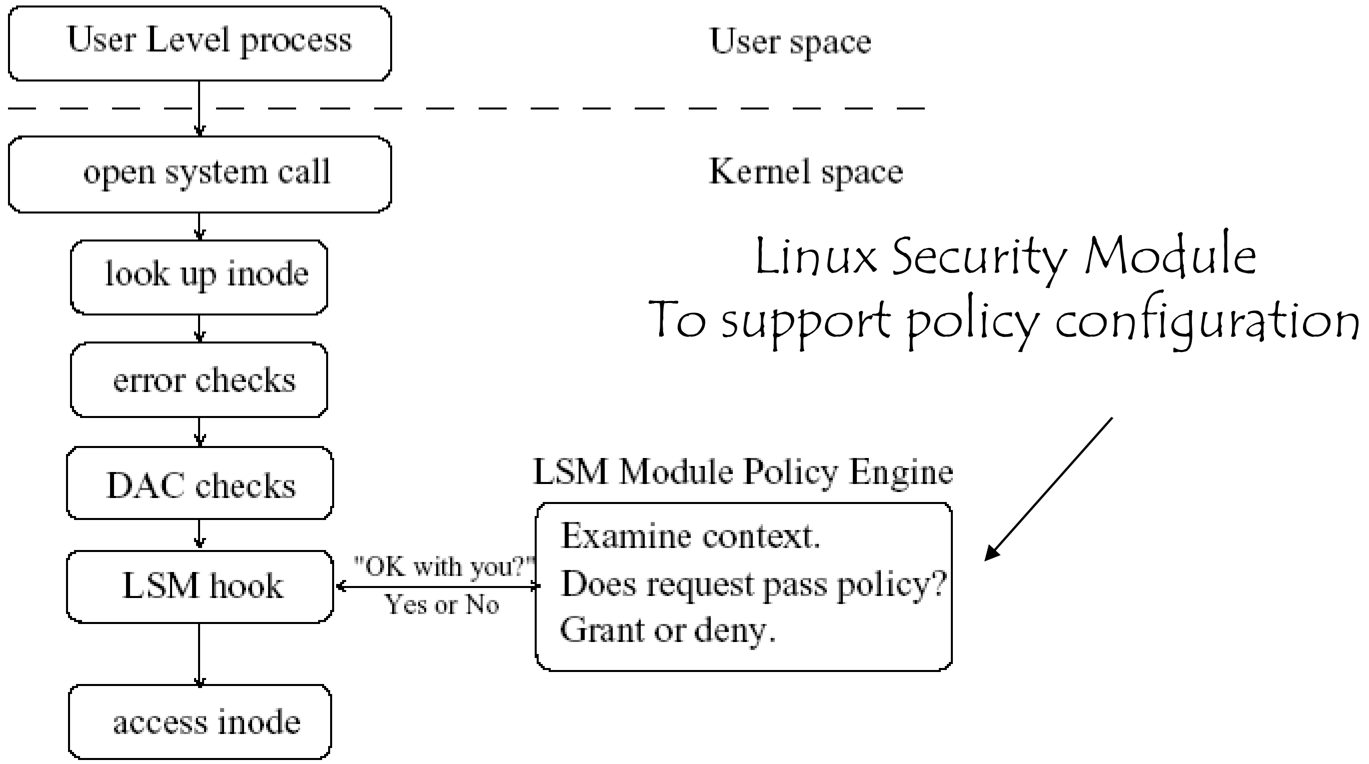
- MLS portion of Security Context is composed of 4 parts
  - Low/High
  - Sensitivity/Category
- Includes syntax to define dominance of security levels
- Subjects with range of levels considered *trusted subjects*
- Implements a variation of Bell-La Padula

Architecture

LSM

- Kernel framework for security modules
- Provides a set of hooks to implement further checks
- Usually placed after existing DAC checks and before resource access
- Implications? SELinux check is not called if the DAC fails
- Makes auditing difficult at times.

SELinux - Implementation



User Level process

User space

open system call

Kernel space

look up inode

error checks

DAC checks

LSM hook

"OK with you?"

Yes or No

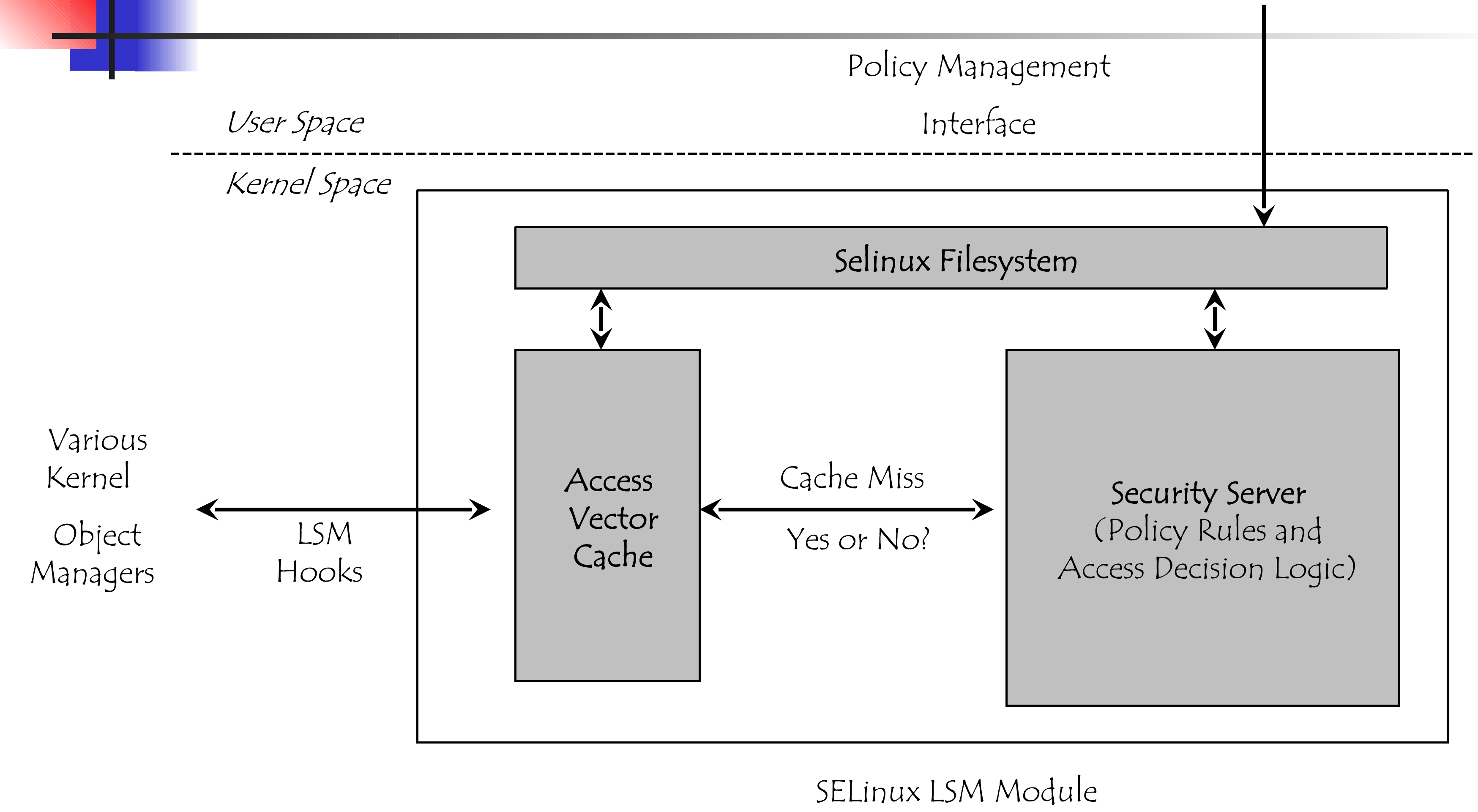
access inode

LSM Module Policy Engine

Examine context.
Does request pass policy?
Grant or deny.

Linux Security Module
To support policy configuration

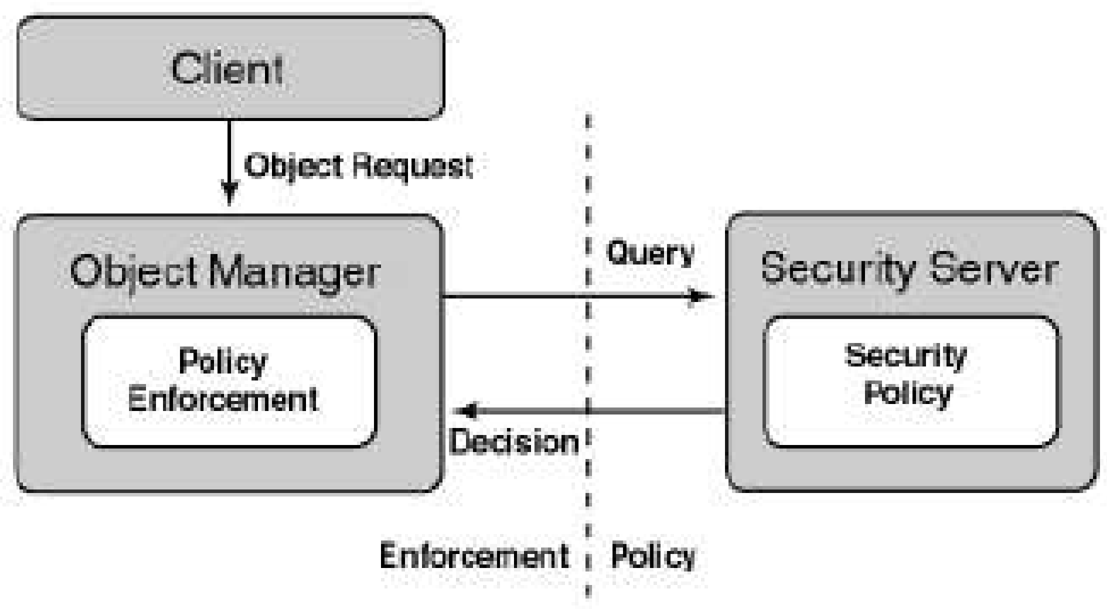
SELinux LSM Module



General Model - PSID

- PSID = SID for persistent object
- Each file system includes a file to map each inode into a PSID and then into a context
- This file is used when the file system is mounted

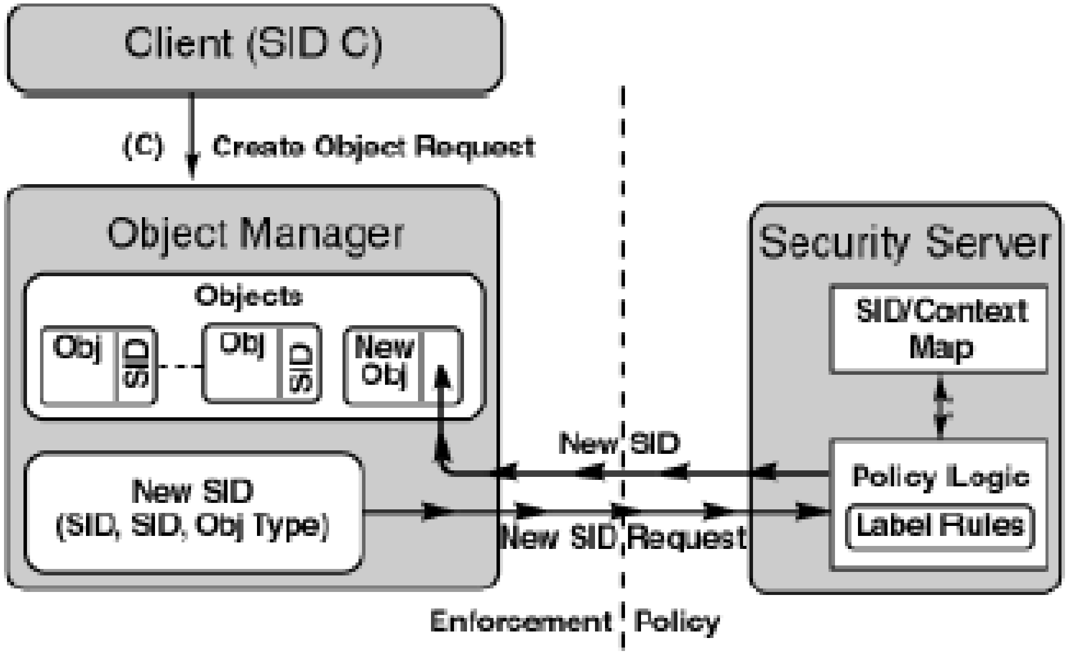
General model - Interactions



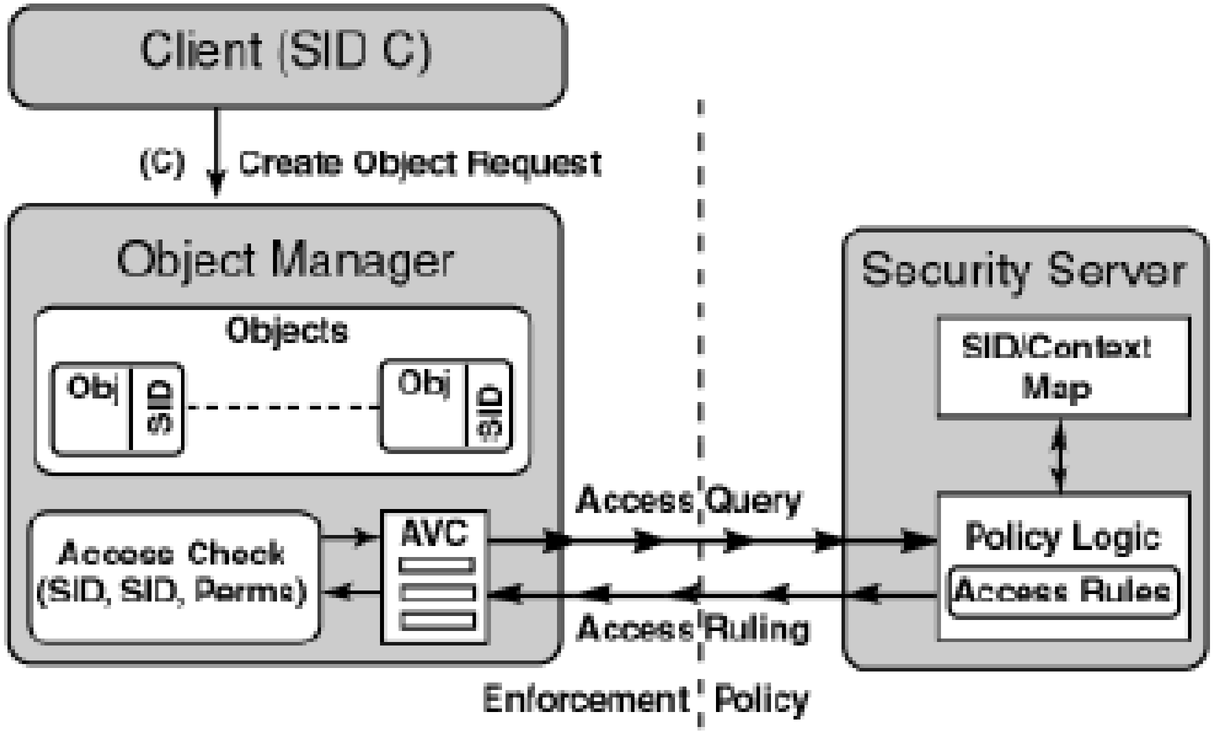
Enforcement with no informatio about the security policy

Security policy with no enforcement

SID and Context

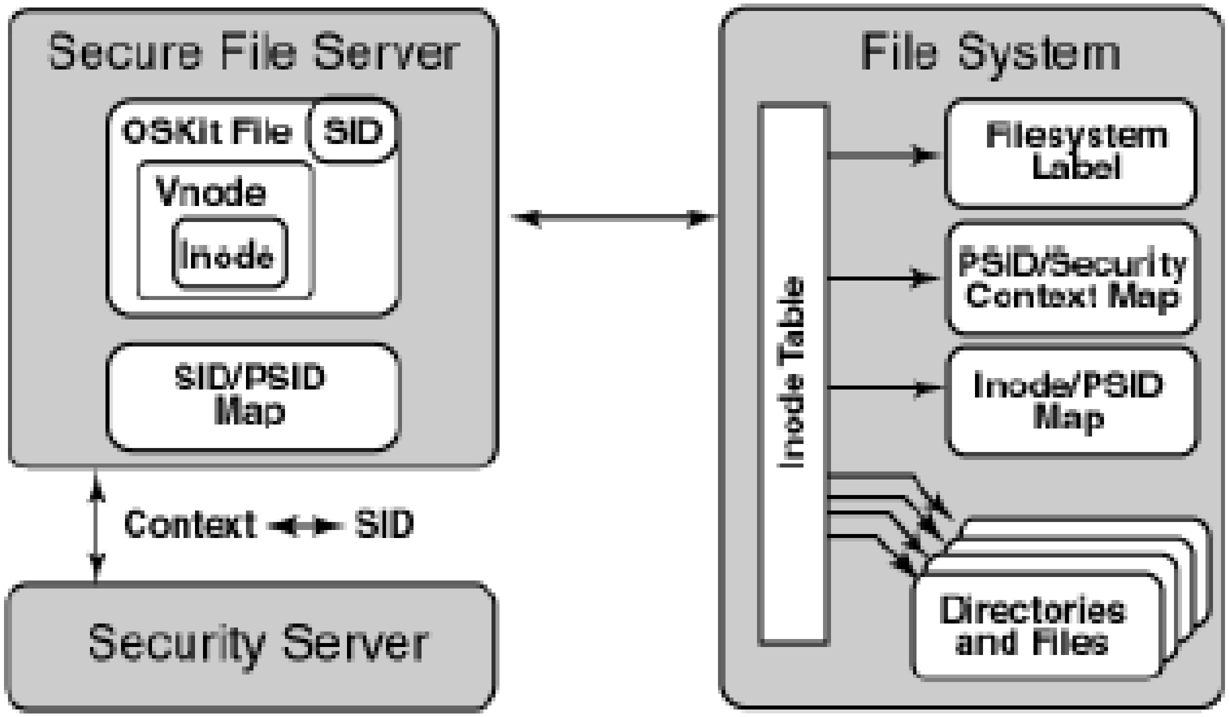


Caching

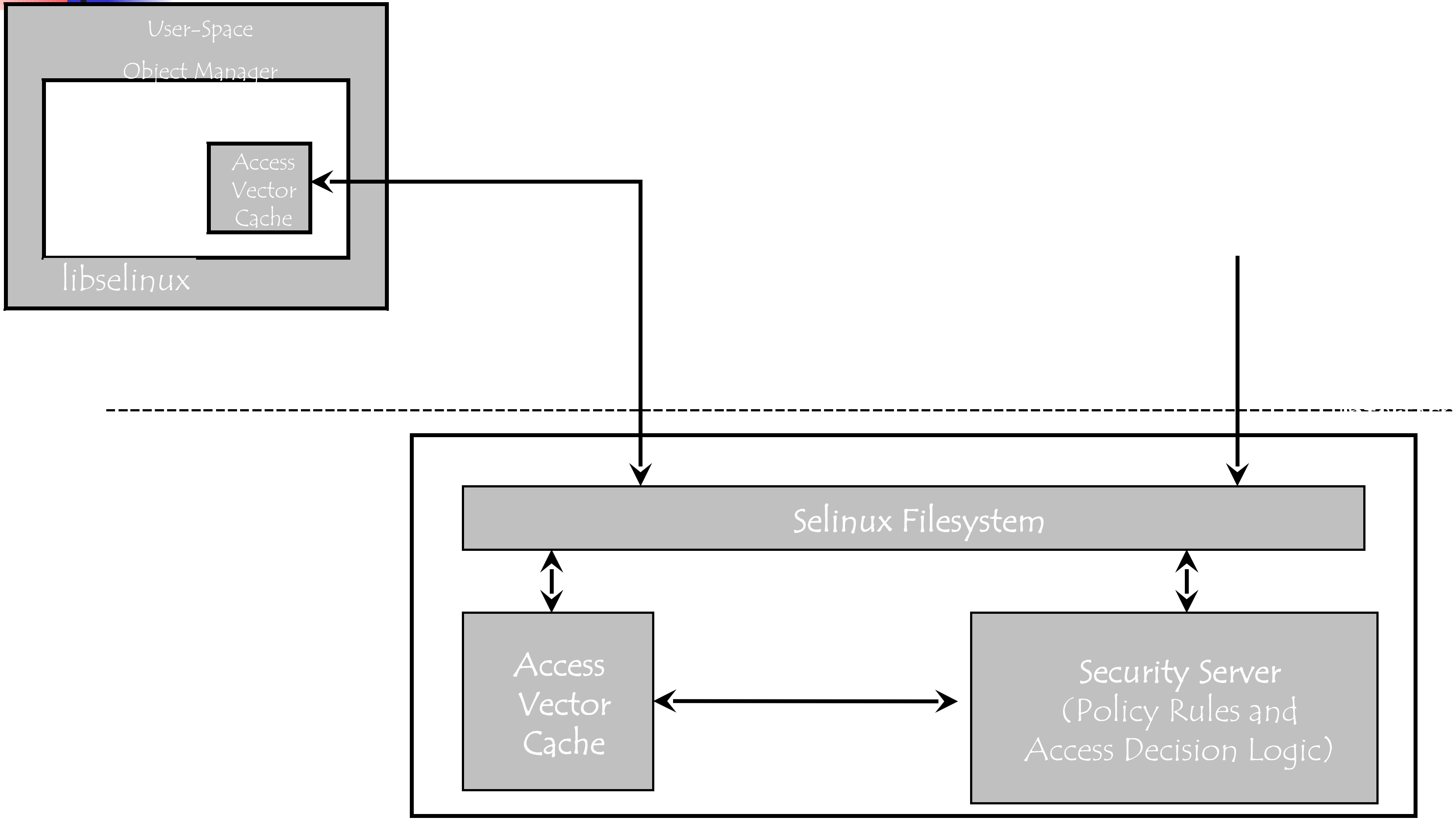


We reduce security to reduce the overhead

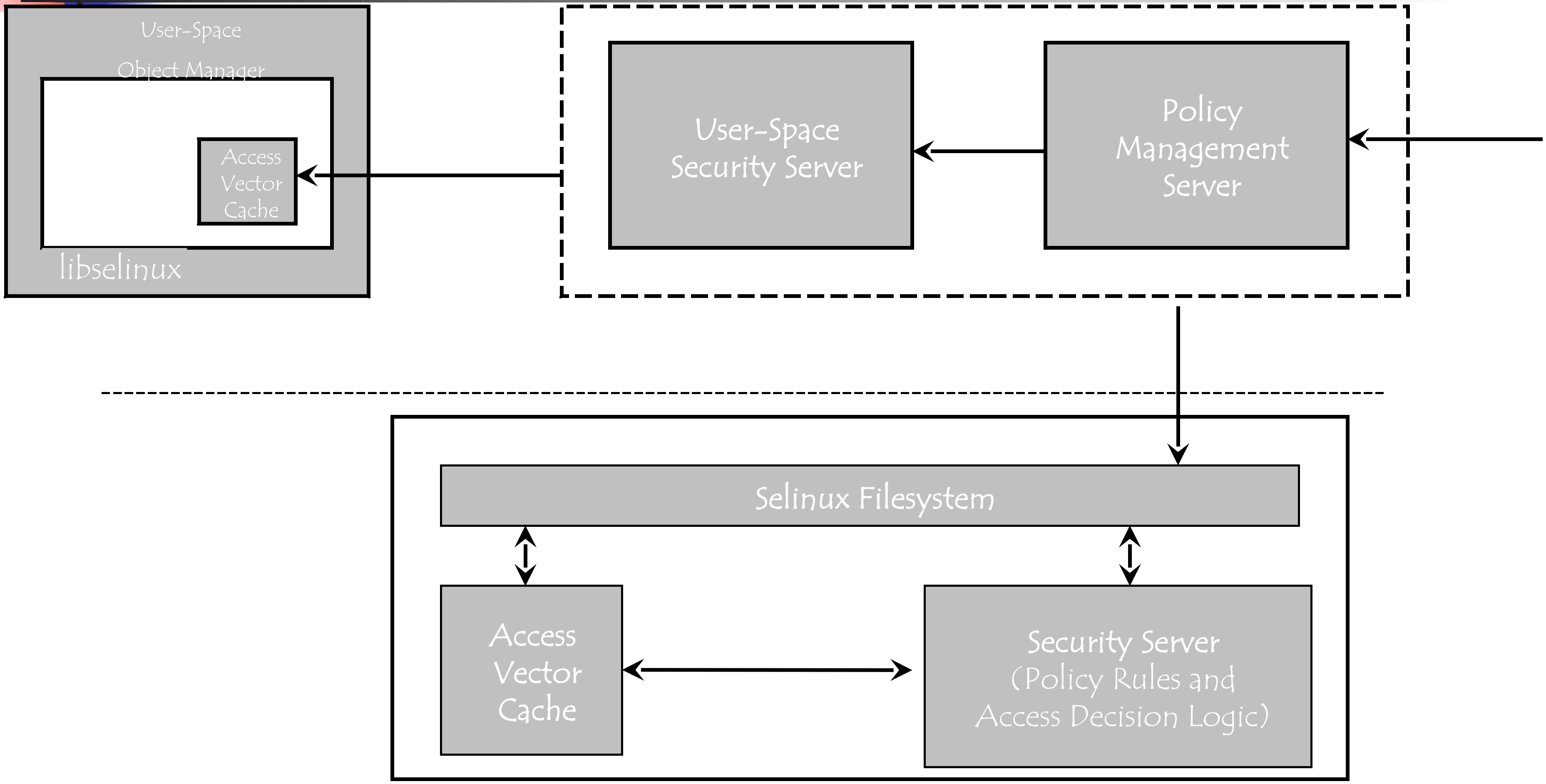
PSID


Secure File Server
OSKit File
SID
Vnode
Inode
SID/PSID Map
Context ←→ SID
Security Server
File System
Inode Table
Filesystem Label
PSID/Security Context Map
Inode/PSID Map
Directories and Files

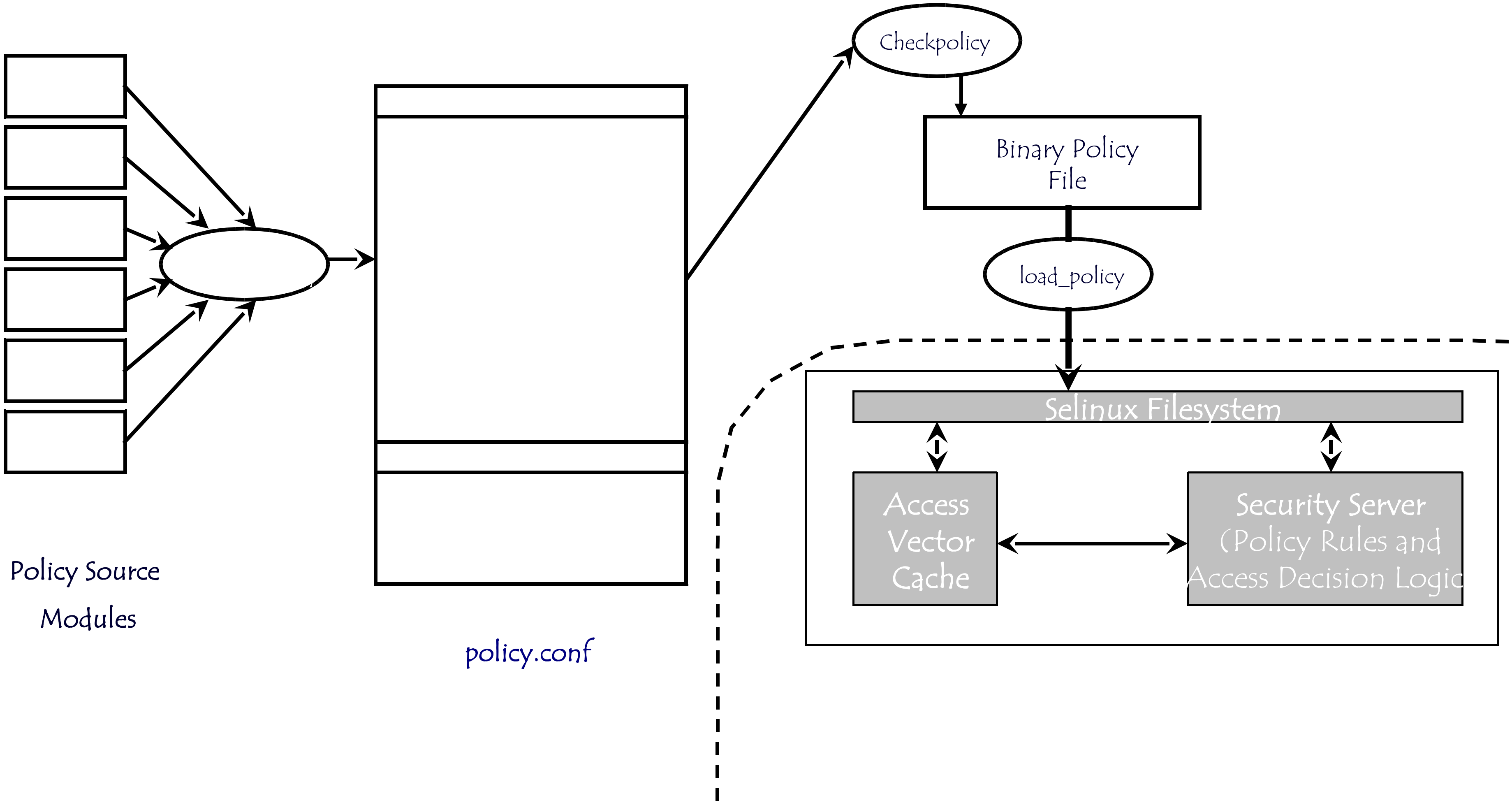
Userspace Object Managers



Policy Server



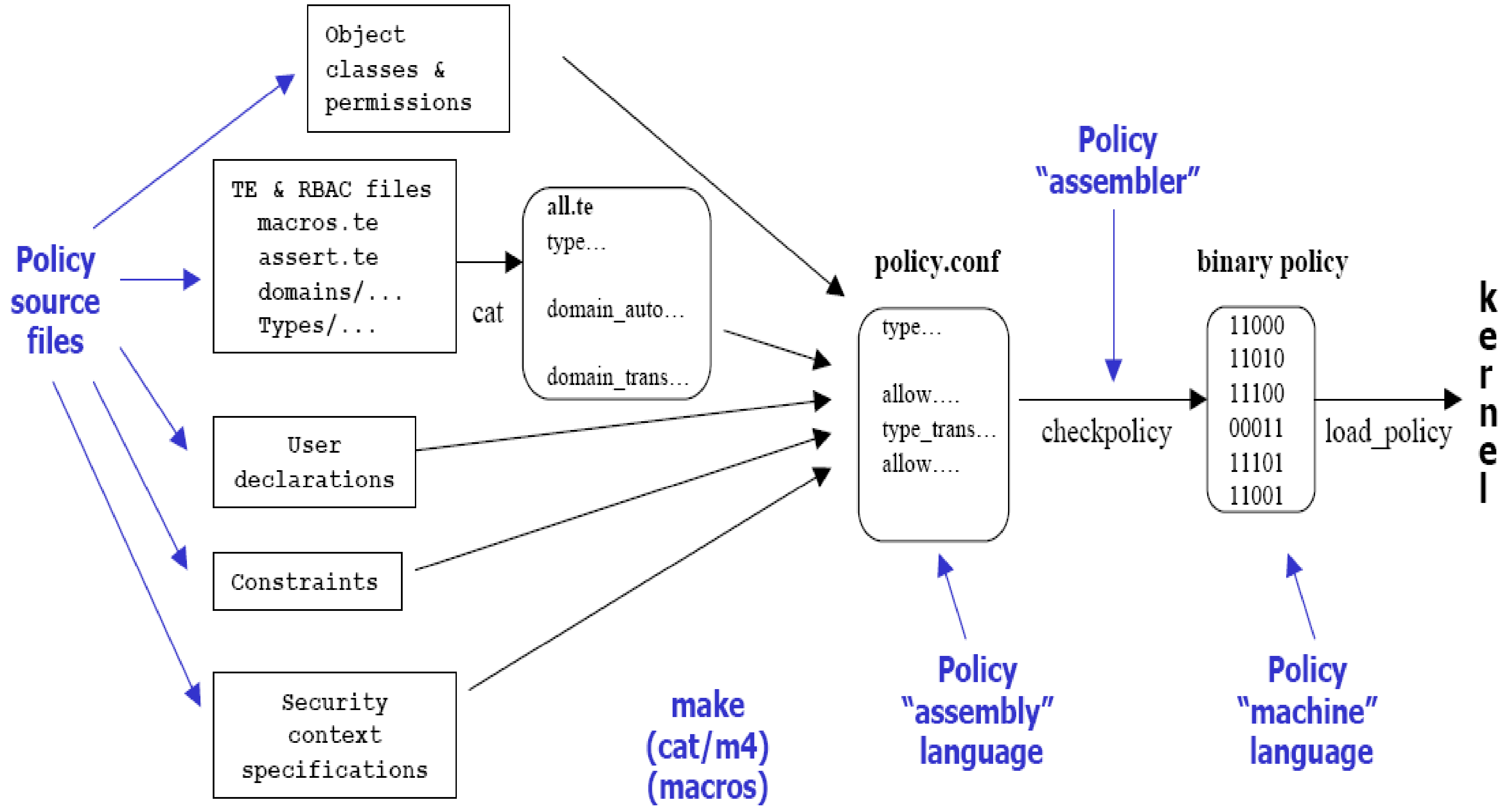
Policy Language



Policy Source Modules

policy.conf

SELinux – Policy - Tools

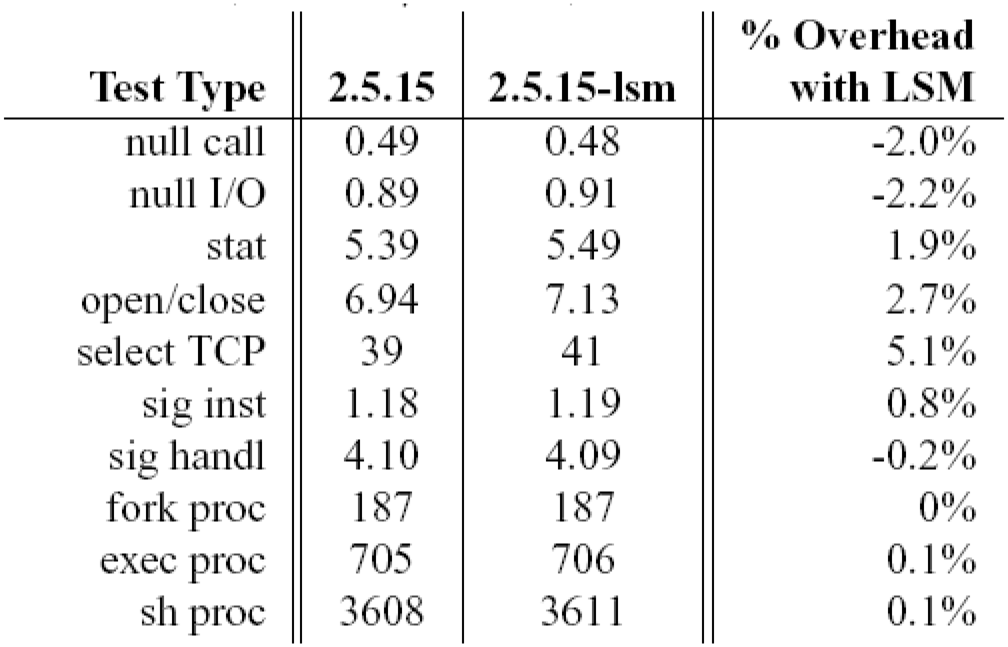


SELinux – Policy

- The description of a policy is rather complex even in the case of simple policies
- As an example, to specify the Linux policy ie to build a SELinux that implements the Linux policy
  - 29 types
  - 121 operations
  - 27.000 rules
- Little support for an high level description and to check the consistency of a policy

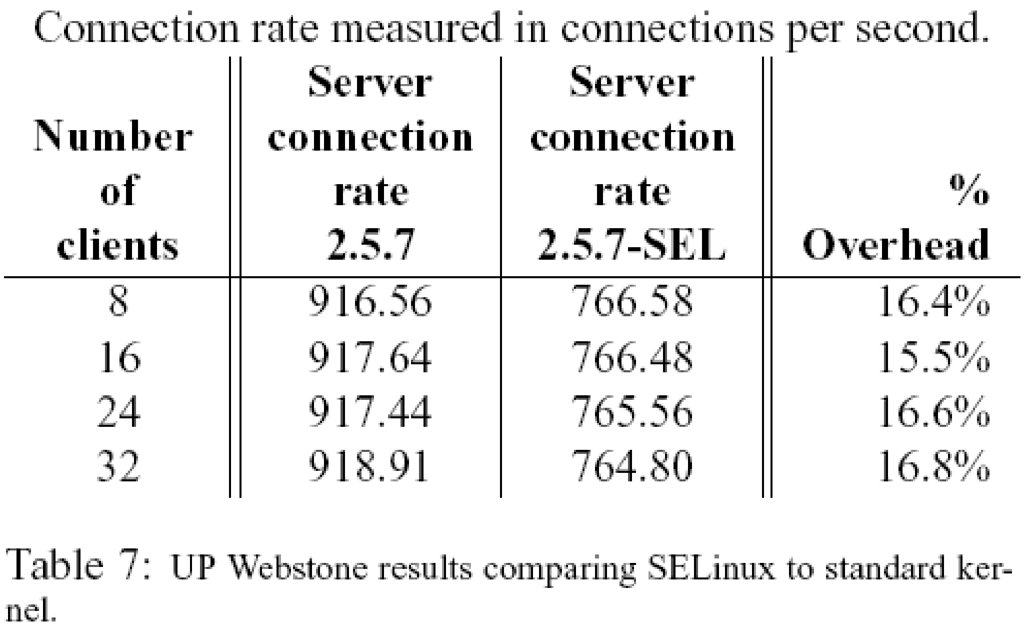
SELinux – Implementation

Implementation of Linux standard Security policy



Test Type	2.5.15	2.5.15-lsm	% Overhead with LSM
null call	0.49	0.48	-2.0%
null I/O	0.89	0.91	-2.2%
stat	5.39	5.49	1.9%
open/close	6.94	7.13	2.7%
select TCP	39	41	5.1%
sig inst	1.18	1.19	0.8%
sig handl	4.10	4.09	-0.2%
fork proc	187	187	0%
exec proc	705	706	0.1%
sh proc	3608	3611	0.1%

Overhead due to SE



Connection rate measured in connections per second.

Number of clients	Server connection rate 2.5.7	Server connection rate 2.5.7-SEL	% Overhead
8	916.56	766.58	16.4%
16	917.64	766.48	15.5%
24	917.44	765.56	16.6%
32	918.91	764.80	16.8%

Table 7: UP Webstone results comparing SELinux to standard kernel.

This points out that the cost is

- Acceptable if we consider the execution overhead
- Fairly large if we consider the complexity of the description

Webstone

Creates a load on a Web server by simulating multiple clients which can be thought of as users, Web browsers that retrieves files from a Web server. This simulation is carried out using multiple Web clients running on one or more computers. It is possible to run in excess of 100 simulated Web clients on a single computer.

In order to create large loads on a Web server, WebStone is able to distribute Web clients among client computers. The Webmaster is the program that controls all of the testing done by WebStone. It can be run on one of the client computers or on a separate computer. The Webmaster distributes the Web client software and test configuration files to the client computers. The Webmaster combines the performance results from all the clients into a single summary report.

AppArmor

It pairs a program with a profile and it supplements rather than replaces the default Discretionary Access Control (DAC). It's impossible to grant a process more privileges than it had in the first place.

SELinux attaches labels to all files, processes and objects and is very flexible. However configuring SELinux is very complicated and requires a supported filesystem.

AppArmor on the other hand works using file paths and its configuration can be easily adapted.

AppArmor

It proactively protects the operating system and applications from external or internal threats and even zero-day attacks by enforcing a specific rule set on a per application basis.

Security policies completely define
- what system resources individual applications can access,
- with what privileges.

Access is denied by default if no profile says otherwise.

Default policies are included with AppArmor.

Every breach of policy triggers a message in the system log and with real-time violation warnings popping up on the desktop.

Profile Modes

AppArmor can operate in two types of profile modes:

Enforce

In the enforce mode, system begins enforcing the rules and report the violation attempts in syslog or auditd and operation will not be permitted.

Complain

In the complain mode, system doesn't enforce any rules. It will only log the violation attempts.

Profile

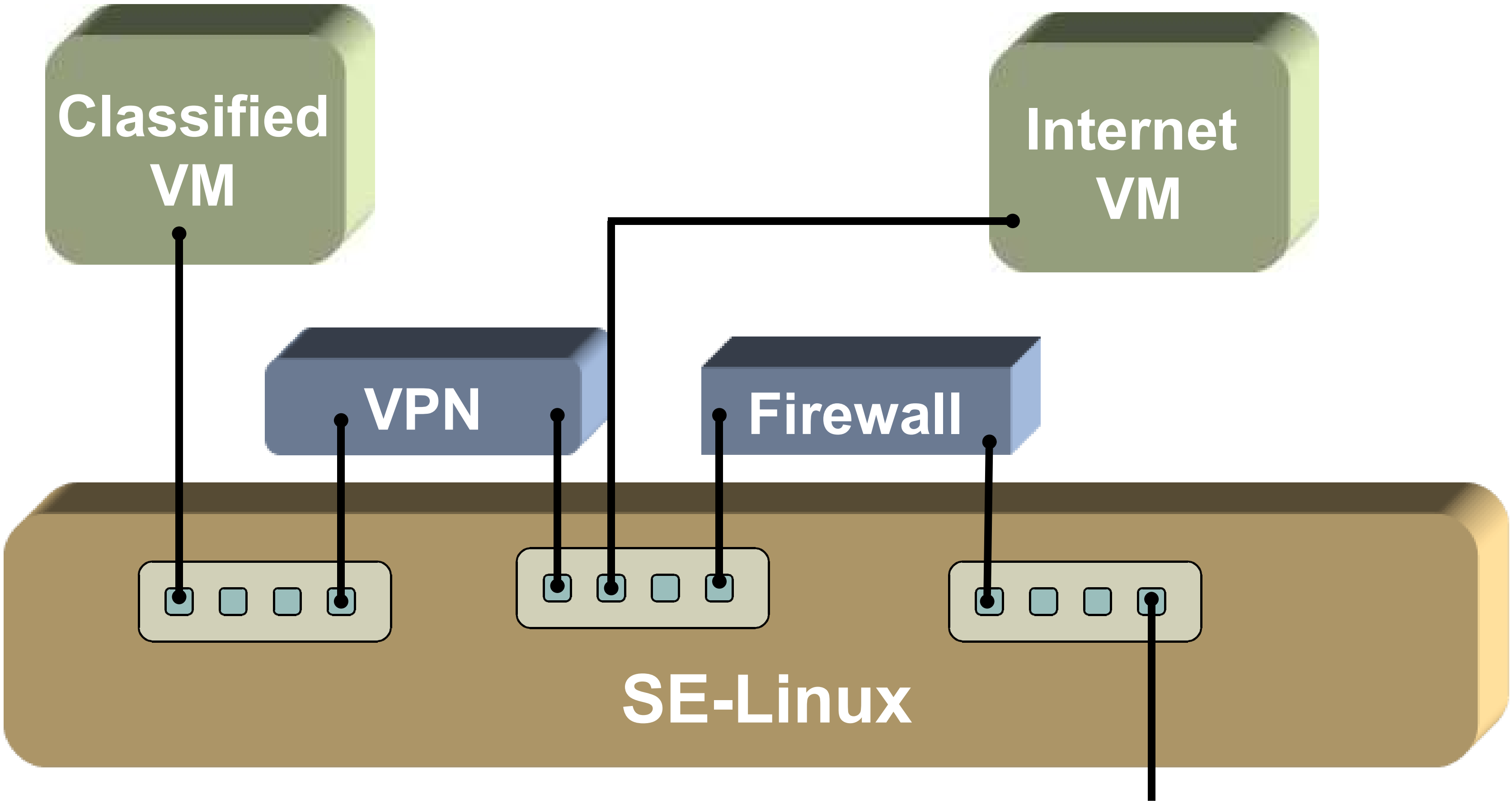
```
/usr/sbin/mysqld {
  #include <abstractions/base>
  ...
  capability dac_override,
  capability sys_resource,
  capability setgid,
  capability setuid,
  network tcp,
  /etc/hosts.allow r,
  /etc/hosts.deny r,
  /etc/mysql/*.pem r,
  /etc/mysql/conf.d/ r,
  /etc/mysql/conf.d/* rw,
  /etc/mysql/*.cnf r,
   }
```

Path entries: This has information on which files the application is allowed to access. Rights are read, write, execute, lock

Capability entries: determines the privileges (posix capability) a confined process is allowed to use.

Network entries: determines the connection-type. For example: tcp. For a packet-analyzer network can be raw or packet etc.

Using SELinux - NSA NetTop



NetTop = SE-Linux + VMware

- SE-Linux:
  - Security-Enhanced Linux
  - Mandatory Access Control with flexible security policy
- VMware Workstation:
  - VMs configuration limited by security policy
- NetTop:
  - Locked-down SE-Linux policy
  - No networking on the host itself

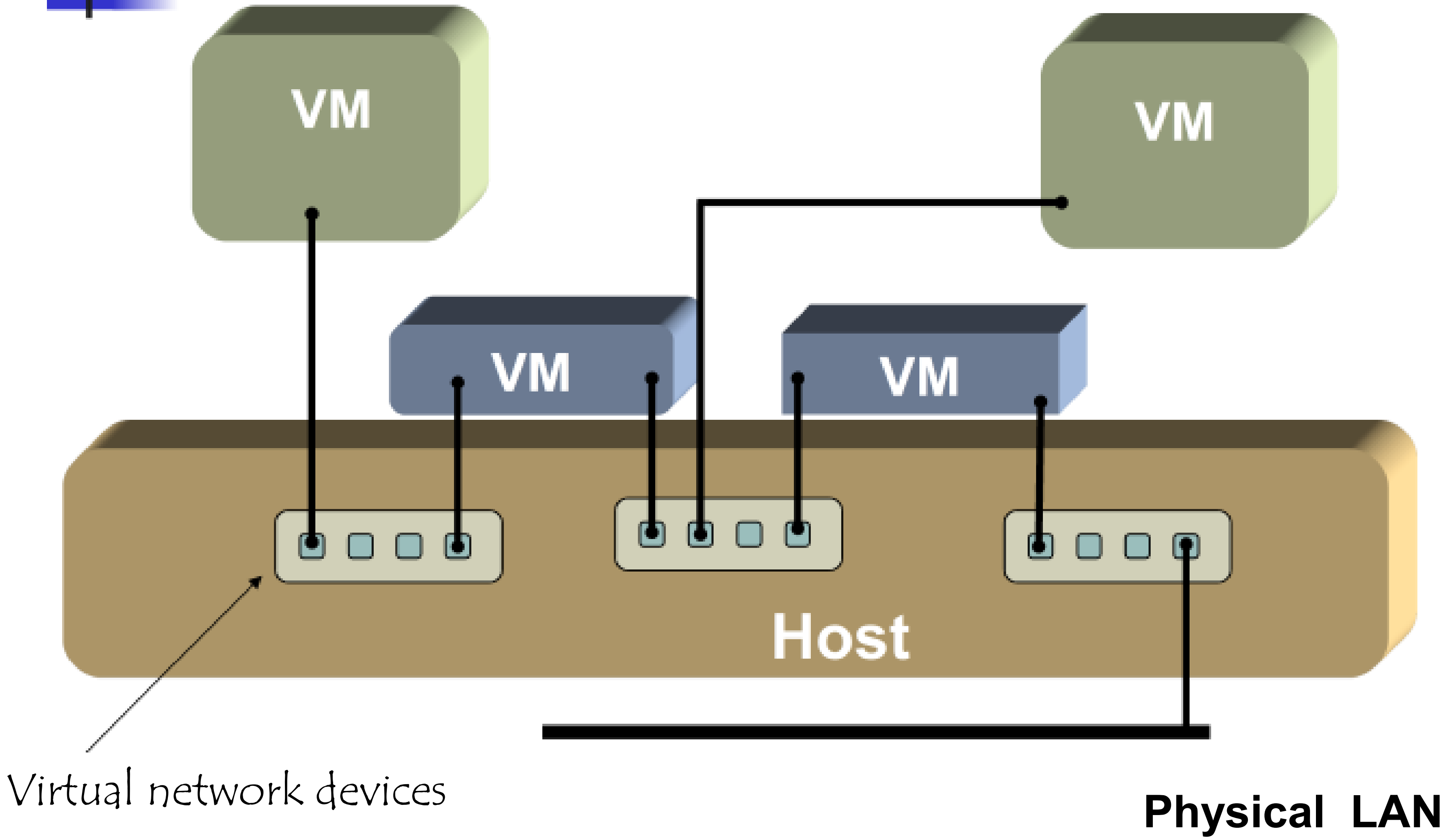
Attributes of VMware Virtual Machines

- Software compatibility
  - Runs pretty much all software
  - BIOS, OS, Apps, viruses, …
- Near-native performance
- Encapsulation
  - Virtual machines are not tied to physical machines
- Consolidation
  - Run multiple VMs on a single desktop or server
- Isolation

Isolation at multiple levels

- Data :
  - Each VM is managed independently
  - Different OS, disks (→ files, registry), MAC address (→ IP address)
  - Data sharing is not possible = Each file system is a SE Linux file
- Faults:
  - Crashes are contained within a VM
- Performance
  - Guaranteed performance levels for individual VMs
- Security
  - No assumptions on the software running inside a VM.

Flexible Networking: VMnets



Mandatory Interposition on all I/O

- 2 levels of mandatory I/O interposition (VM level and OS level)
- Guest cannot directly initiate I/O
  - →All guest I/O operations mediated by VMware
- VMware relies on the host for I/O access
  - → VMware process uses system calls to execute all I/O requests.
- Example: networking, disk I/O

Processes running on the Host system

- “See without being seen” advantage
  - Very difficult within a computer
  - Possible on the host
- Observation points:
  - Networking (through vmnet)
  - Disk I/O (read and write)
  - Any other I/O
  - Physical Memory of the VM

Why NetTop?

Example: Access to classified networks

- Traditional tension : Security vs. Usability
  - Secure systems are not that usable
    - E.g: require some particular OS setups
  - Flexible systems are not that secure
    - Many documented examples
- Additional requirement:
  - Data cannot flow between networks of different classification
- Conventional solution:
  - Dedicate distinct computer for access to each network

Security of Isolation

- Q: How securely isolated are the virtual machines?
- A: Pretty well …

NATIONAL SECURITY AGENCY
UNITED STATES OF AMERICA

All together now …

We have seen a large number of countermeasures

- Static = to be adopted before attacks occur, hopefully before deploying the system, (the sooner, the better)
  - firewall
  - VPN
  - ids
  - new OS
  - honeypot
- Dynamic = in general they tune the behavior of the security policy to better react to an ongoing attack. Distinct cost/perfomance ratio may be acceptable when under attack

But …

- Any approach based upon a risk analysis faces the problem of partial and unaccurate information on threats, their resources, their potential impact
- Even when considering partial information some problems cannot be solved as exemplified by "know unknown vs unknown unknown)
- Hybrid threat are arising (attacking with more weapons ie cyber attack and fake news etc., the Gerasimov strategy)
- Resilience is the new goal

Problems (in the words of one of my friends)

“Emerging cyber realities and technologies are presenting new threats with uncertain intensity and frequency and the vulnerabilities and consequences in terms of the extent of casualties, economic losses, time delays, or other damages are not yet fully understood or modeled. As a result, risk calculations become more uncertain and generate costly solutions since multiple, often hypothetical, threat scenarios could point to many vulnerabilities and catastrophic system failures that are unaffordable to mitigate, absorb, or recover” (do you remember black swan?)

Resilience (in the words of the same friend)

Security, robustness and risks are connected, they are focused on preventing system from degrading and keeping functionality within acceptable level before and after the adverse event. Resilience is a very different concept. Oxford defines it as

"the capacity to recover quickly from difficulties."

A resilience assessment thus starts with the assumption that system is affected, functionality is impaired and is focused on evaluation of recovery speed.

Resilience

Cyber resilience is the ability of an ICT infrastructure

- to resist to a stress (robustness)
- to reconfigure its ICT structure to offer some services when resistence is no longer possible (reconfiguration)
- to return to a normal behavior after the stress ends (elasticity)

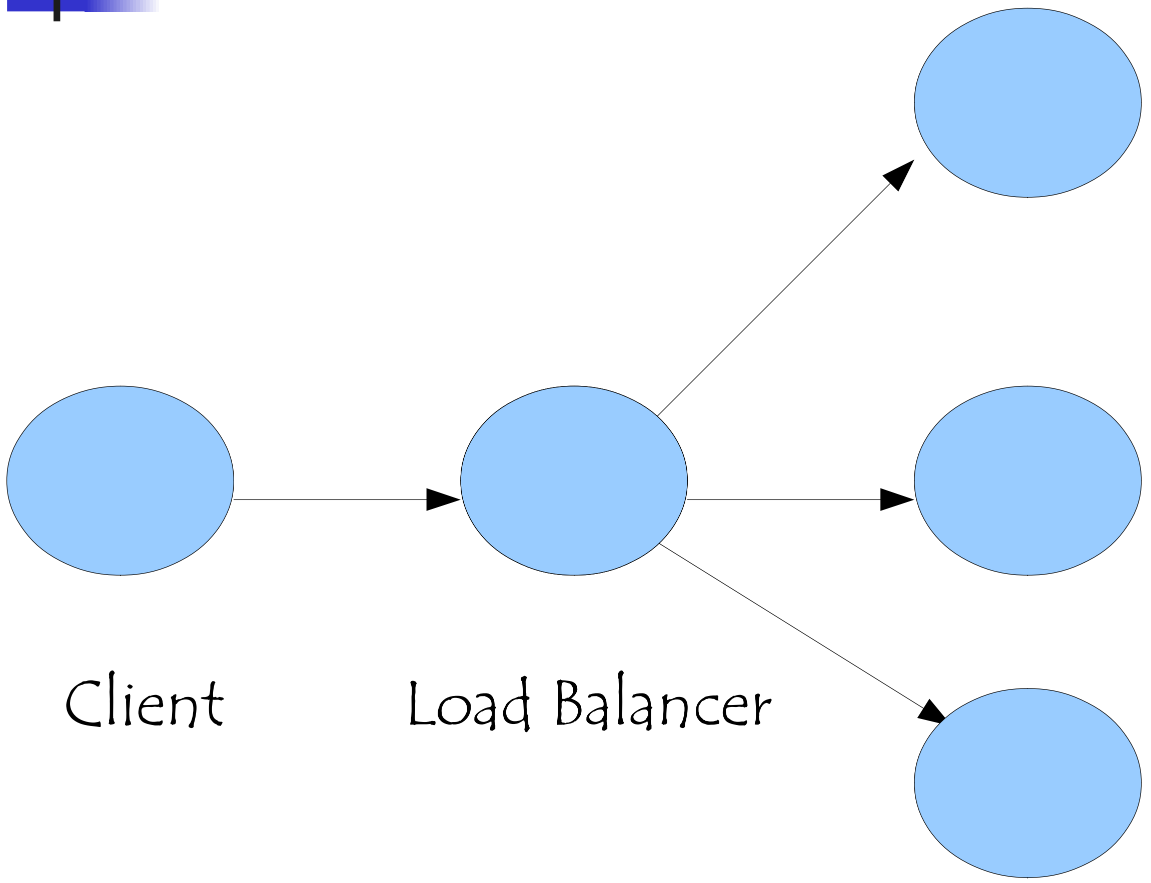
Cyber resilience

- While a robust system does not change its behavior and the services it offers even when under attack a resilient one reconfigures and focuses on critical services
- Robustness may turn into fragility
  - due to unexpected events
  - with wrong threat modelling
- Resilience= Robustness now and when new threats will arise

Cyber resilience

- Some important steps to achieve resilience
  - Identify key threats and assess their impact on critical cybersystems and functions
  - Increase robustness
  - Classify and prioritize critical services.
  - Set cyber-resilience goals and objectives for critical services
- Any resilience requires some redundancy we need to choose a compromise between the investment in robustness and the one in redundancy and diversity

Redundancy + Diversity



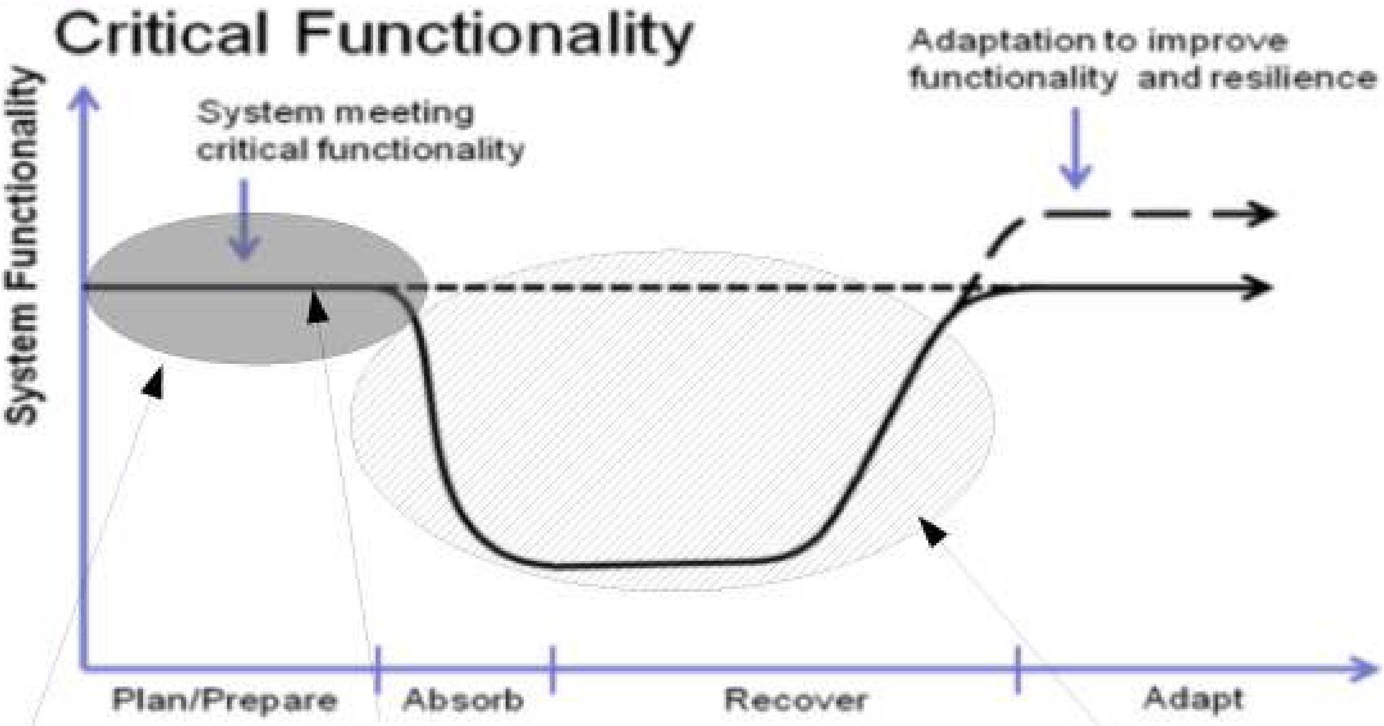
Redundancy + Diversity

Google advocates this model even for releasing a new release or a new version of a server

Cyber resilience

- Some important steps to achieve resilience
  - Identify key threats and assess their impact on critical cybersystems and functions
  - Increase robustness
  - Classify and prioritize critical services.
  - Set cyber-resilience goals and objectives for critical services
- Any resilience requires some redundancy we need to choose a compromise between the investment in robustness and the one in redundancy and diversity

Resilience


Critical Functionality
System Functionality
System meeting critical functionality
Adaptation to improve functionality and resilience
Plan/Prepare
Absorb
Recover
Adapt

Attack/Crash
Natural event

Robustness aims to maximise this time

Resilience aims to minimize the area under the curve

Resilience

- Plan/Prepare:
  - robustness for know unknow and known
  - redundancy/reconfiguration for unknow unknow
- Absorb
  - at first robustness,
  - then redundancy
  - then reconfiguration
- Recover return to a normal behavior, maybe with distinct performances