

Parte 6

Protocolli di Sicurezza storici

Accesso singolo

Protocollo di sicurezza – esempio 1

- Def.** Usare unica credenziale di autenticazione per
■ Dovuto a Needham-Schröder, 1978
accedere a tutti i servizi
- Presuppone una PKI con crittografia perfetta

- Soluzione comoda ma poco robusta

– Un'unica 1. Alice → Bob : $\{Alice, N_a\}K_{bob}$



Alice



2. Bob → Alice : $\{N_a, N_b\}K_{alice}$

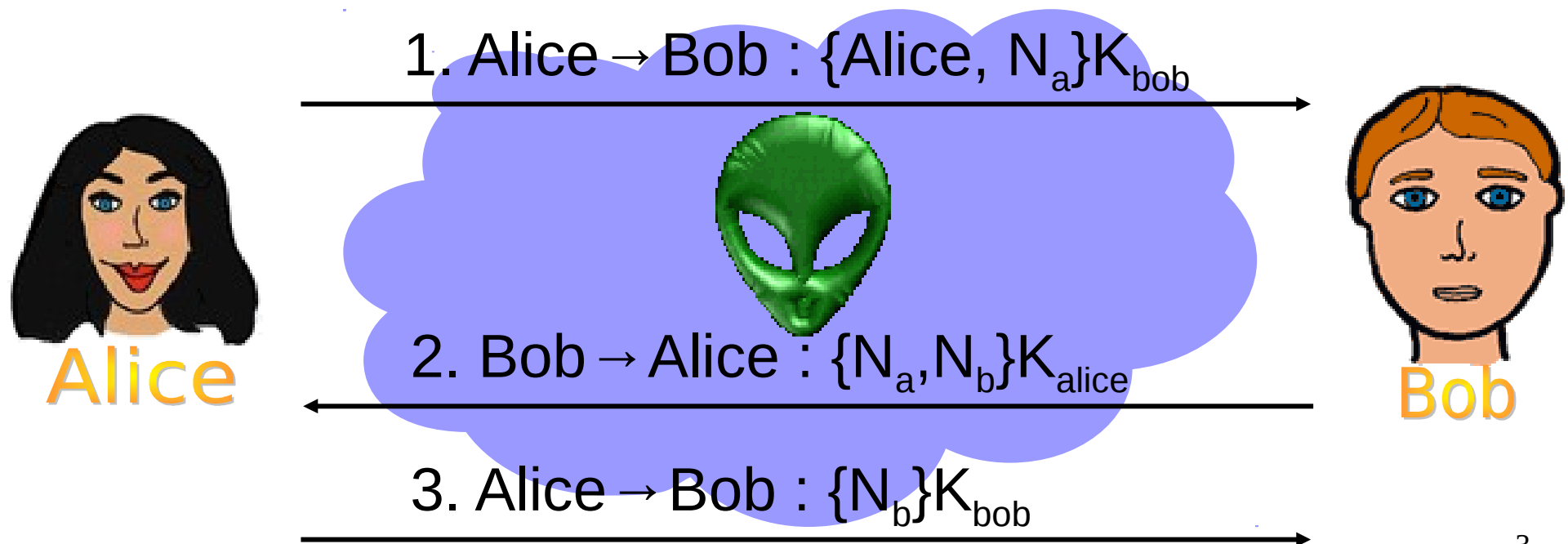


Bob

3. Alice → Bob : $\{N_b\}K_{bob}$

Protocollo di sicurezza – esempio 1

- Dovuto a Needham-Schröder, 1978
- Presuppone una PKI con crittografia perfetta



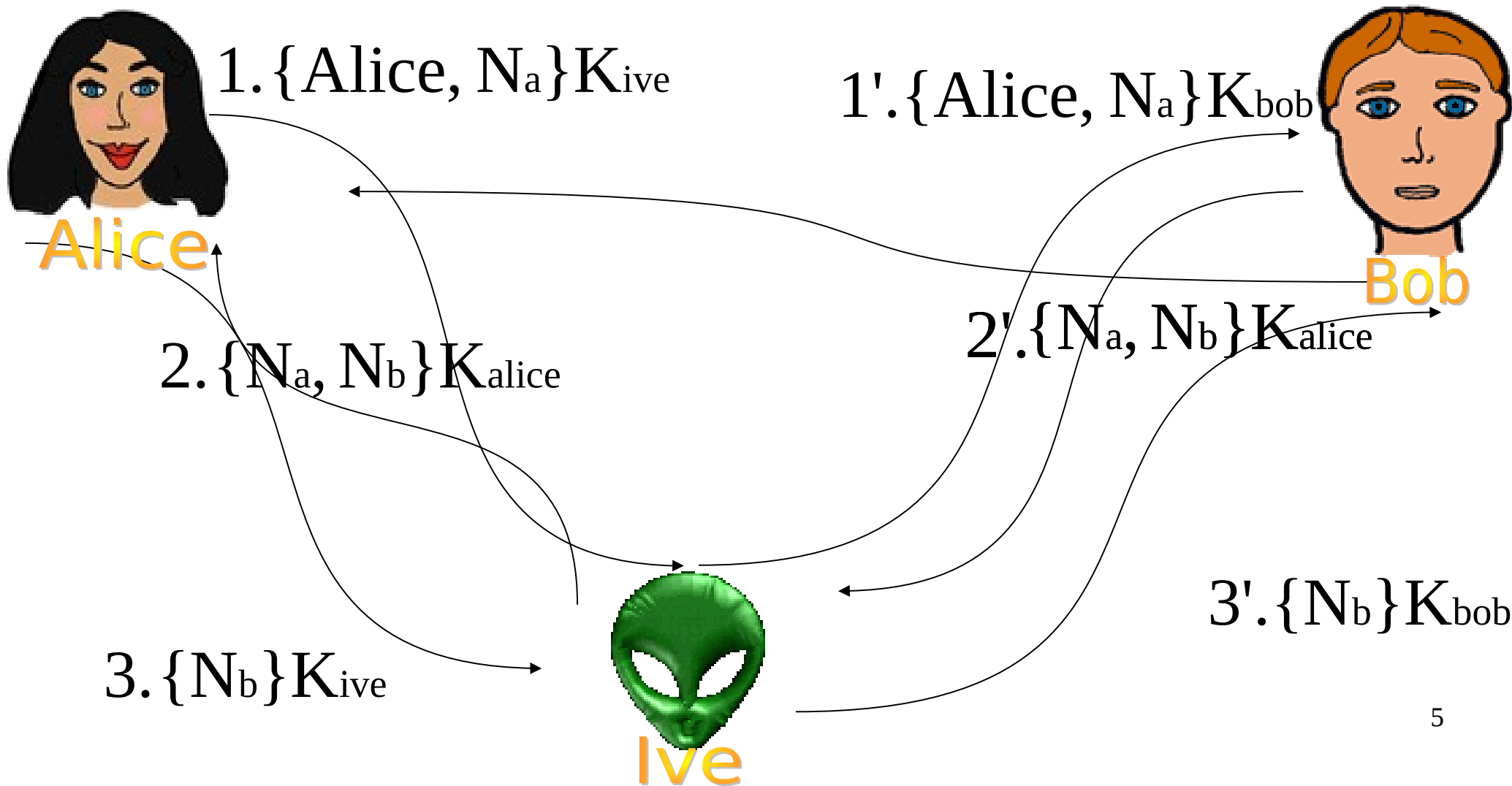
Obiettivi di sicurezza del protocollo (goal)

1. Alice \rightarrow Bob : $\{Alice, N_a\}K_{bob}$
2. Bob \rightarrow Alice : $\{N_a, N_b\}K_{alice}$
3. Alice \rightarrow Bob : $\{N_b\}K_{bob}$

1. Autenticazione reciproca degli utenti
 - ☐ Etichette mittente e ricevente inaffidabili!
 - ☐ Autenticazione garantita da segretezza delle nonce
2. Segretezza delle nonce scambiate

Gli obiettivi falliscono!

1. Alice \rightarrow Bob : $\{Alice, N_a\}K_{bob}$
2. Bob \rightarrow Alice : $\{N_a, N_b\}K_{alice}$
3. Alice \rightarrow Bob : $\{N_b\}K_{bob}$



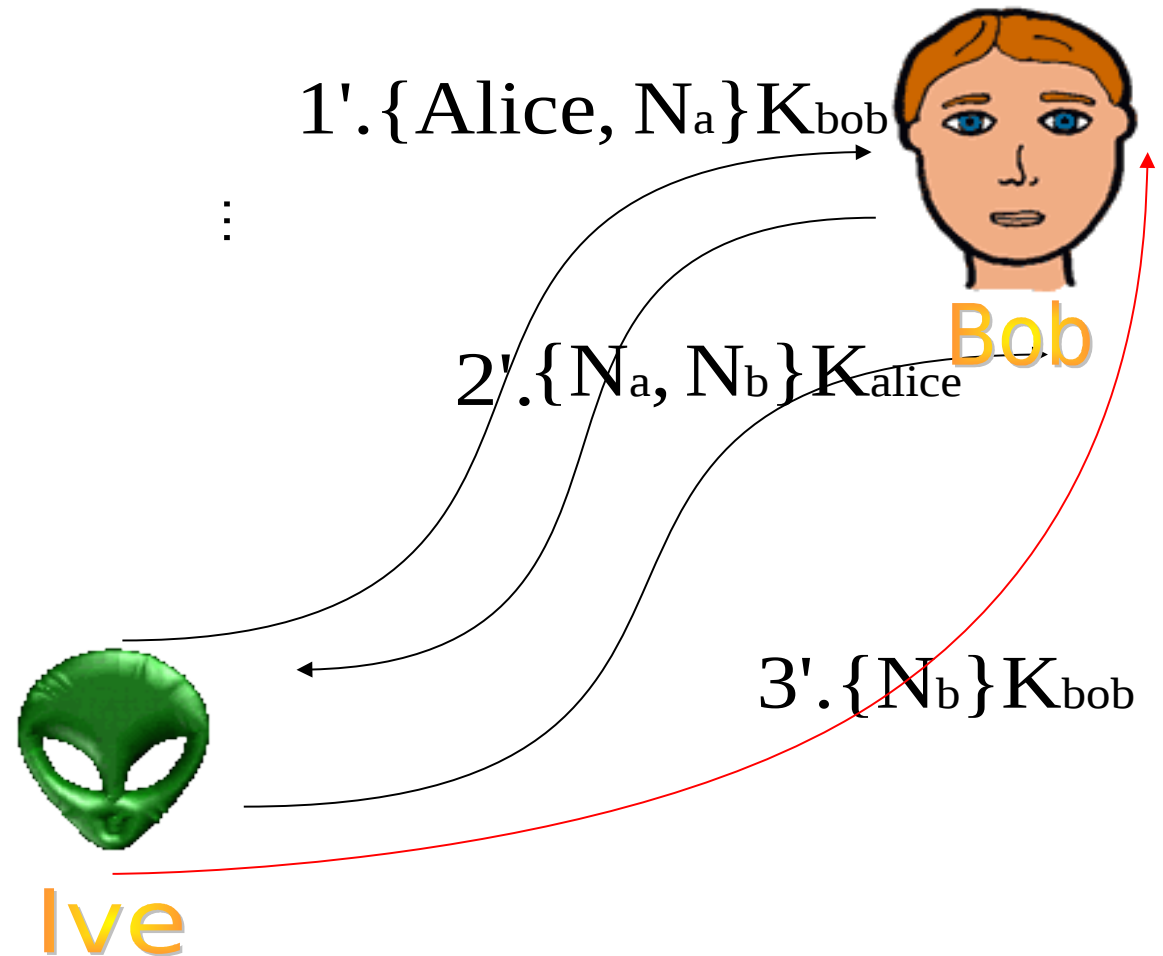
Gli attacchi visti (attacco di Lowe, 1995)

- 2 sessioni **interlacciate**
- Nell'ipotesi che Alice cominci con la spia
- Attivi, da **posizione intermedia**
 - Segretezza di N_b fallisce col passo 3
 - Autenticazione di Alice con Bob fallisce col passo 3'. Come??
- Sicurezza (segretezza, autenticazione) fallita anche nell'ipotesi di crittografia perfetta!!

Conseguenze dell'attacco

- Se Bob fosse una banca e gli altri due correntisti...

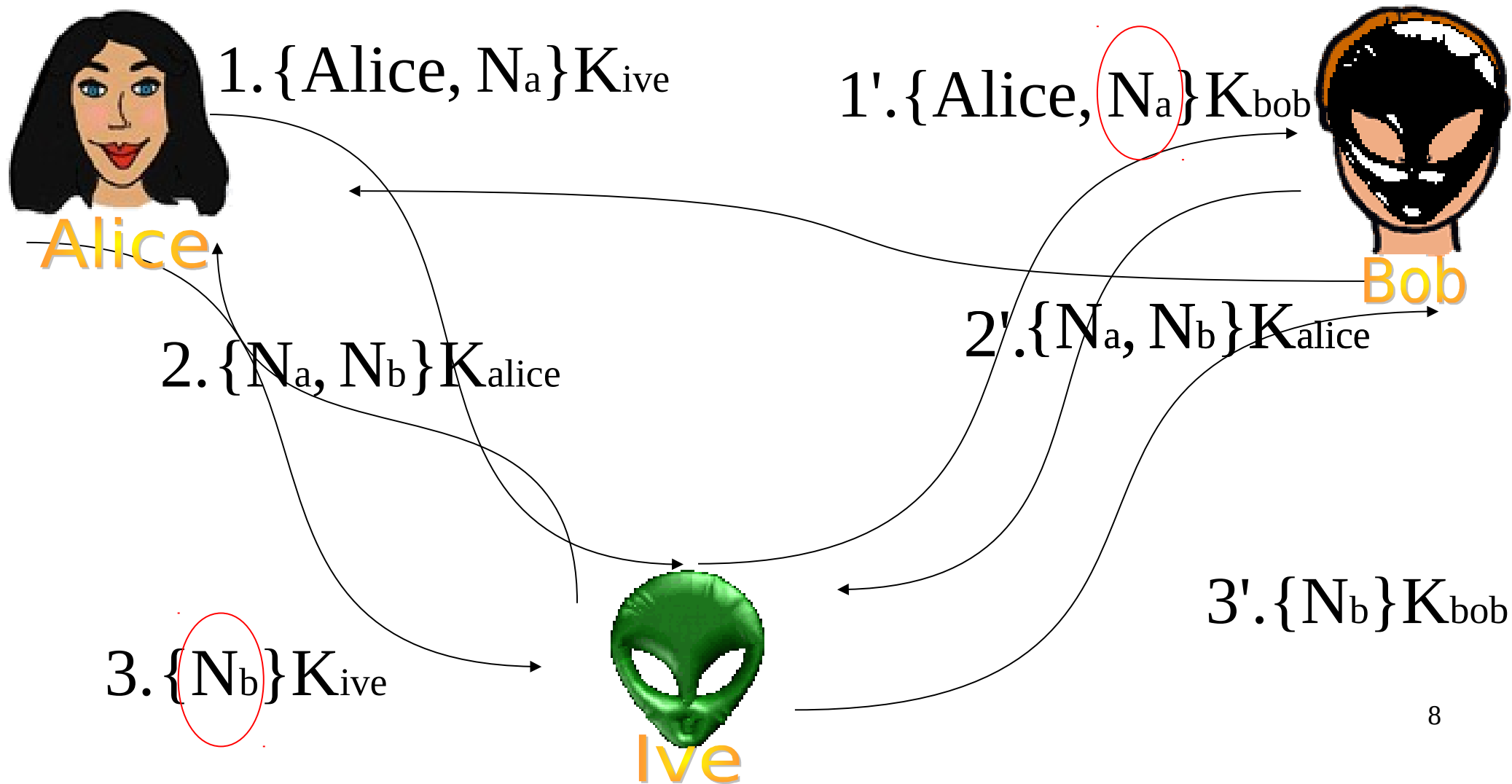
- Se Alice fosse il docente e gli altri due studenti...



4'. $\{N_a, N_b, \text{"trasferisci 10000€ dal conto di Alice al conto di Ive"}\} K_{bob}$

4'. $\{N_a, N_b, \text{"l'esame di domani è cancellato"}\} K_{bob}$

Lo stesso attacco studiato in GA



Vendetta nel modello GA

- Ipotesi: Bob scopra l'importanza di N_a
- Se anche Alice è una banca, Bob può vendicarsi su Ive come segue

5'. $\{N_a, N_b, \text{"trasferisci 20000 € dal conto di Ive al conto di Bob"}\}$ Kalice

Protocollo di sicurezza – esempio 2

- Dovuto a Woo-Lam, metà anni '80
- Usa crittografia simmetrica
- Usa un **TTP** (Trusted Third Party),
che possiede un
database di tutte
le chiavi
- Goal: autentica di
Alice con Bob

1. $A \rightarrow B : A$

2. $B \rightarrow A : N_b$

3. $A \rightarrow B : \{N_b\}K_a$

4. $B \rightarrow TTP : \{A, \{N_b\}K_a\}K_b$

5. $TTP \rightarrow B : \{N_b\}K_b$

Un attacco su Woo-Lam

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_b$
3. $A \rightarrow B : \{N_b\}K_a$
4. $B \rightarrow TTP : \{A, \{N_b\}K_a\}K_b$
5. $TTP \rightarrow B : \{N_b\}K_b$

- B vede indietro N_b
- Pertanto autentica l'utente cui l'ha associata, ossia A
- A potrebbe perfino essere off-line
- B non distingue la sessione!

1. $C \rightarrow B : A$

1'. $C \rightarrow B : C$

2. $B \rightarrow A : N_b$

2'. $B \rightarrow C : N_b'$

3. $C \rightarrow B : \{N_b\}K_c$

3'. $C \rightarrow B : \{N_b\}K_c$

4. $B \rightarrow TTP : \{A, \{N_b\}K_c\}K_b$

4'. $B \rightarrow TTP : \{C, \{N_b\}K_c\}K_b$

5. $TTP \rightarrow B : \{N_b''\}K_b$

5'. $TTP \rightarrow B : \{N_b\}K_b$

I rischi di attacchi aumentano

- 1978: Needham-Schröder, 6 pagine
- Metà anni '90: **SSL**, 80 pagine
- Fine anni '90: **SET**, 1000 pagine!

Quasi vent'anni per scoprire che un protocollo di 6 pagine celava un bug! Allora...

Potenziali soluzioni??

- Needham-Schröder asimmetrico:
 - ? 1. Alice \rightarrow Bob : $\{\{Alice, Na\}K_{alice^{-1}}\}K_{bob}$
 - ? 1. Alice \rightarrow Bob : $\{\{Alice, Na\}K_{alice^{-1}}\}K_{bob}$
2. Bob \rightarrow Alice : $\{\{Na, Nb\}K_{bob^{-1}}\}K_{alice}$
 - ? 2. Bob \rightarrow Alice : $\{\{Na, Nb\}K_{bob^{-1}}\}K_{alice}$
 - ? 2. Bob \rightarrow Alice : $\{Na, Nb, Bob\}K_{alice}$
 - ? 1. Alice \rightarrow Bob : $\{Alice, Bob, Na\}K_{bob}$

Potenziali soluzioni??

- Woo-Lam:

? 3. $A \rightarrow B : \{A, Nb\}Ka$

? 5. $TTP \rightarrow B : \{A, Nb\}Kb$

? 4. $B \rightarrow TTP : \{A, \{A, Nb\}Ka\}Kb$

? 2. $B \rightarrow A : Nb, B$

Principi di disegno: explicitness

Def. Se le identità del mittente e del ricevente sono significative per il messaggio, allora è prudente menzionarle esplicitamente

Problema: quando sono “significant”??

Principi di disegno: explicitness

Def. Se le identità del mittente e del ricevente sono significative per il messaggio, allora è prudente menzionarle esplicitamente

Problema: quando sono “significant”??

Symmetric Needham-Schröder

1. $A \rightarrow TTP : A, B, N_a$

2. $TTP \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}K_b\}K_a$

3. $A \rightarrow B : \{K_{ab}, A\}K_b$

4. $B \rightarrow A : \{N_b\}K_{ab}$

5. $A \rightarrow B : \{N_b - 1\}K_{ab}$

The diagram includes two red boxes: one around the term $\{K_{ab}, A\}K_b$ in step 3, and another around the term $\{K_{ab}, A\}K_b$ within the larger package in step 2. A red arrow labeled "ticket" points from the box in step 2 to the box in step 3.

- A che serve N_a ?
- K_{ab} è **chiave di sessione**
- Mutua autentica mediante passi 4 e 5

Replay attack

Def. Spacciare informazione (*chiavi,...*) obsoleta, magari violata, come recente

- Supponiamo che C abbia violato una vecchia chiave di sessione K_{ab} che B condivide con A

...

3. $C \rightarrow B : \{K_{ab}, A\}K_b$ (rispedito identico)

4. $B \rightarrow A : \{N_b'\}K_{ab}$ (intercettato)

5. $C \rightarrow B : \{N_b'-1\}K_{ab}$

- B autenticerebbe A e quindi accetterebbe di usare K_{ab}



Parte 9

IPSec



Protocolli di sicurezza

- Client/server non commerciali: *Kerberos*
- E-mail: *PGP, S/MIME*
- Client/server commerciali: *SSL, TLS, SET*
- Connessioni remote: *SSH, SFTP*

Stanno a livelli diversi nello stack di protocolli TCP/IP

Protocolli di sicurezza

- Garantiscono specifiche proprietà di sicurezza mediante l'utilizzo di tecniche specifiche quali

- ☐ Crittografia
- ☐ Steganografia
- ☐ Chaffing & winnowing



- Assumono che un protocollo di comunicazione (trasporto) sia disponibile

Alternative alla crittografia

- Steganografia: nascondere informazioni
 - Può ottenere un buon livello delle proprietà primarie di sicurezza
 - Possiamo scrivere un protocollo di sicurezza steganografico
 - *Cambiare i bit meno significativi di un'immagine (digital watermarking)*

Alternative alla crittografia

- Chaffing & winnowing: mischiare e separare
- Essa stessa una forma di steganografia
 1. Mittente S e ricevente R concordano una chiave k mediante protocollo Diffie-Hellmann
 2. S spedisce a R coppia $m, \text{MAC}(m, k)$
 3. S spedisce a R molte coppie false (chaff) x, y
 4. R seleziona coppia giusta verificandone MAC

Sicurezza a livello di rete

- Per comunicare mediante un protocollo di sicurezza, tutte le macchine devono eseguire un client per quel protocollo
- Se ne può fare a meno se garantiamo sicurezza ad un livello più basso
- Tutte le “vecchie” applicazioni distribuite diventerebbero automaticamente “sicure”



Sicurezza a quale livello

- Precisamente, le proprietà di sicurezza possono essere aggiunte a vari livelli nello stack TCP/IP
- Più in basso andiamo, più la sicurezza diventa trasparente e rigida
- Più in alto andiamo, più la sicurezza necessita apposita gestione e quindi concede più flessibilità

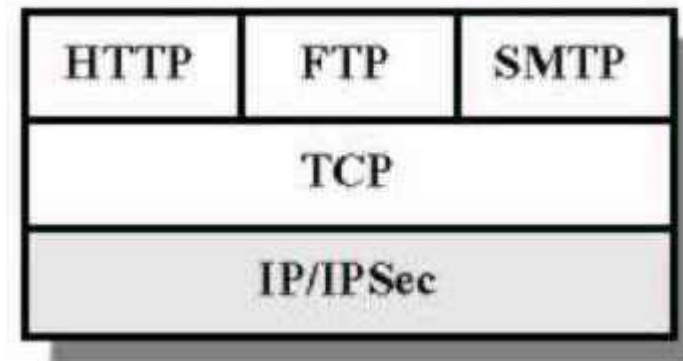
Sicurezza a livello di rete

■ Pro

- ☐ Le applicazioni possono rimanere “ignoranti”
- ☐ Sostanzialmente anche gli utenti

■ Contro

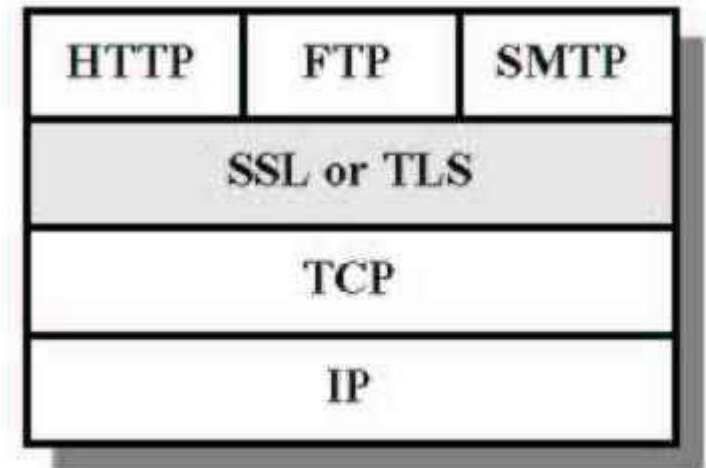
- ☐ Appesantisce notevolmente la comunicazione al punto da facilitare DoS
- ☐ Può richiedere modifiche al S.O.



Sicurezza a livello di trasporto

■ Pro

- In realtà va o al trasporto o alle applicazioni
- I principali browser contengono un client TLS



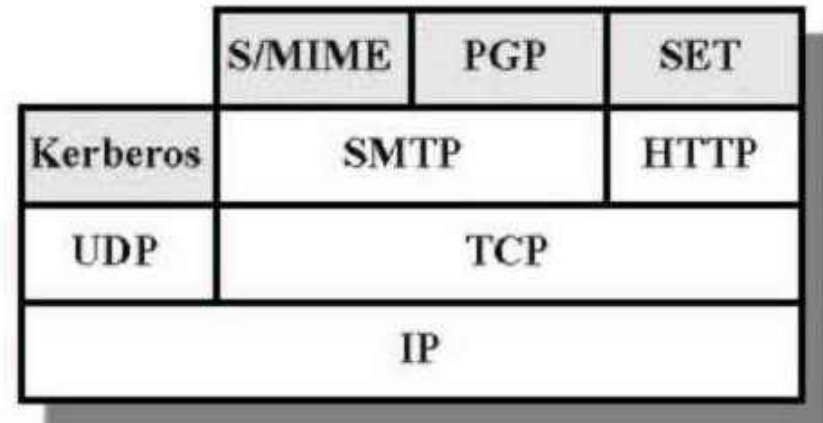
■ Contro

- Ovviamente richiede modifiche (minime) al livello che la riceve

Sicurezza a livello applicazione

■ Pro

- Sicurezza personalizzabile dalle applicazioni



■ Contro

- Disegnare sicurezza a questo livello può essere ingannevole

IPSec

- Suite di protocolli (**AH**, **ESP**, **IKE**) per garantire segretezza autenticazione e integrità a livello IP
- Opzionale per IPv4, mandatorio per IPv6
- Ogni pacchetto IP è arricchito con componenti crittografiche per la sicurezza
- Non tutti i nodi della rete devono essere compatibili IPSec

Security association (SA)

- Relazione unidirezionale fra mittente e destinatario – servono due SA su una comunicazione bidirezionale
- Sancisce il protocollo di sicurezza (AH o ESP) richiesto per quel traffico
- Una SA ha almeno tre campi
 1. **SPI** (Security Parameter Index): per indicizzare
 2. **Indirizzo IP di destinazione**: solo unicast
 3. **Identificatore del protocollo**: AH e/o ESP

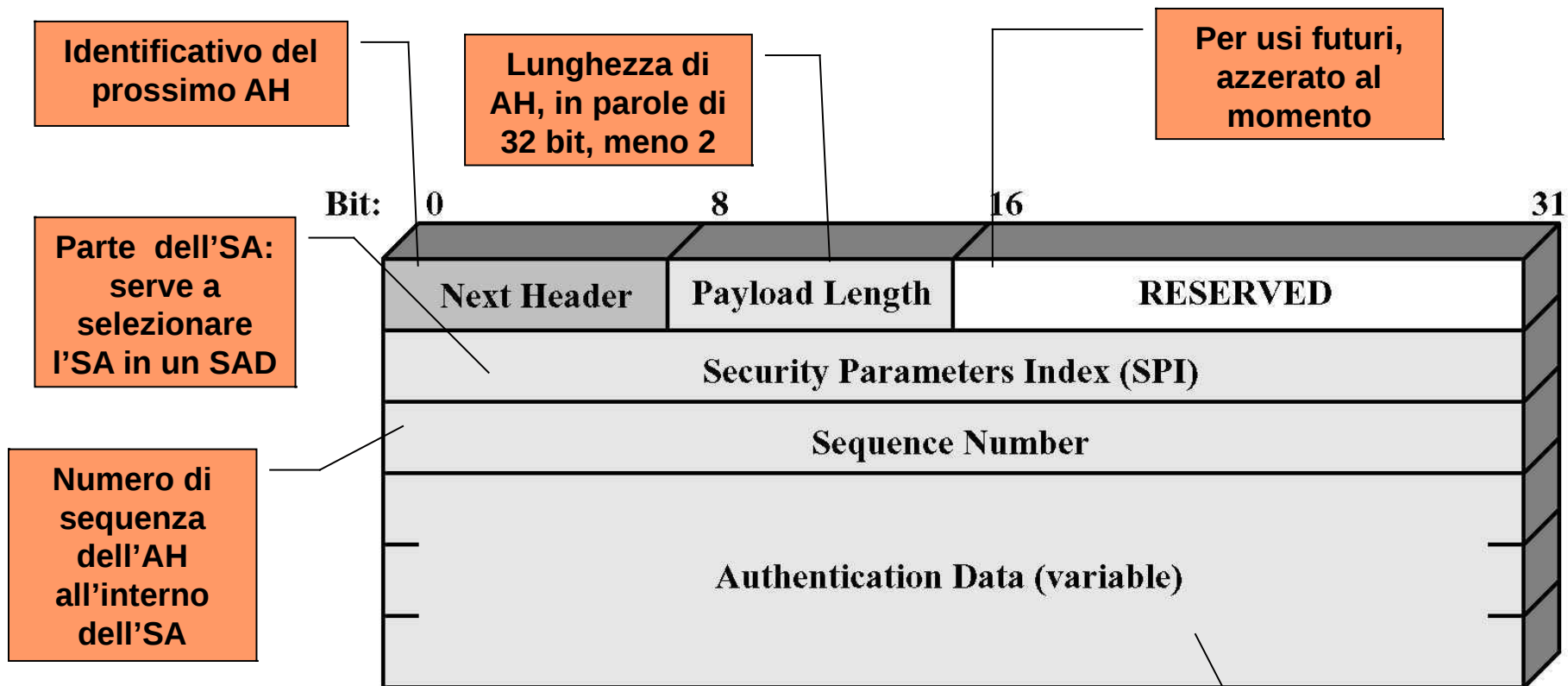
Security assoc. database (SAD)

- Ogni nodo della rete deve mantenere un DB, detto SAD, con tutte le SA attive in quel nodo
- Ogni entry del SAD contiene tutti gli elementi di una SA e in più:
 - 4. **AH info**: info aggiuntive sull'algoritmo di autenticazione
 - 5. **ESP info**: info aggiuntive sull'algoritmo di codifica
 - 6. **Lifetime**: durata della SA

Authentication Header (AH)

- Frammento aggiunto a ciascun pacchetto IP, supporta autenticazione e integrità pacchetti
- Mittente e ricevente abbiano pre-concordato una chiave IKE per poter calcolare MAC
- Autentica l'intero pacchetto esclusi i campi variabili dell'header IP, modificabili dai nodi intermedi (*type of service, flags, fragment offset, time to live, ...*)
- Previene replay attack e IP spoofing

AH – formato



Si evince la lunghezza in bit di ciascun campo.
Qualunque alterazione del pacchetto è rivelata dalla MAC.

MAC del
pacchetto
contenente
questo AH

IPSec – funzionamento di base

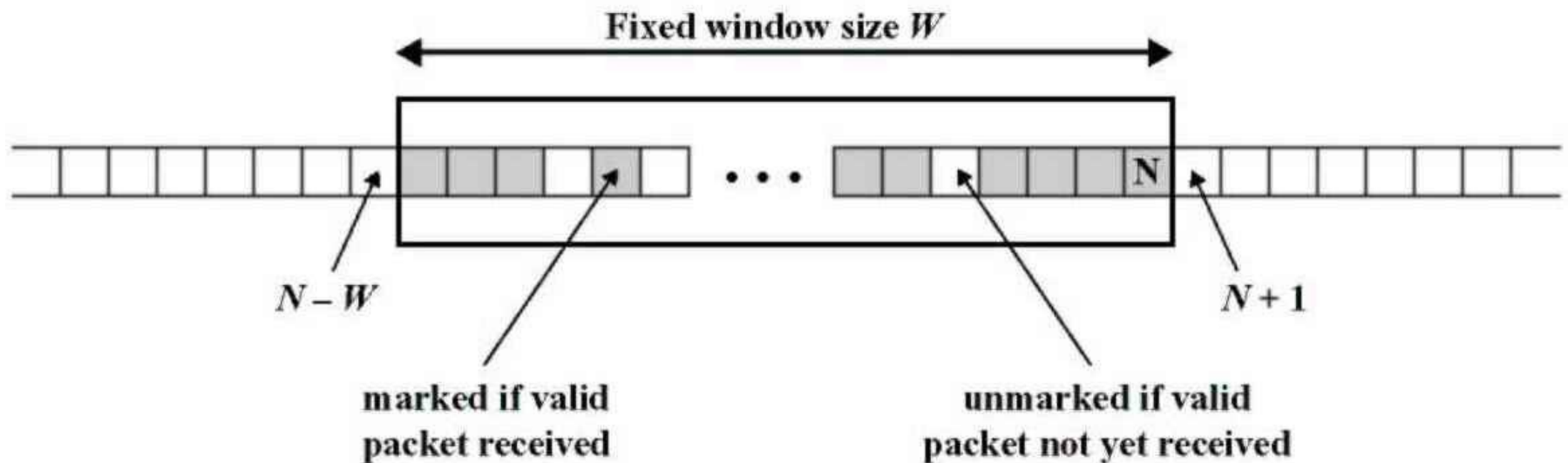
1. Ogni nodo mittente **sceglie** una SA usando il proprio **SPD** (Security Policy Database)
2. L'SPI dell'SA scelta viaggia con l'AH di ciascun pacchetto
3. Visto tale AH, ogni nodo attraversato ne preleva l'SPI per selezionare dal proprio SAD l'SA relativa al pacchetto

Prevenzione dei replay attack

- I pacchetti, anche se autenticati, non possono essere replicati grazie al campo Sequence Number dell'AH
- Il mittente inserisce da 0 a $2^{32}-1$
- Se ha bisogno di altri pacchetti, deve negoziare una nuova SA
- Il ricevente deve scartare i pacchetti
 - vecchi o ripetuti o falsificati

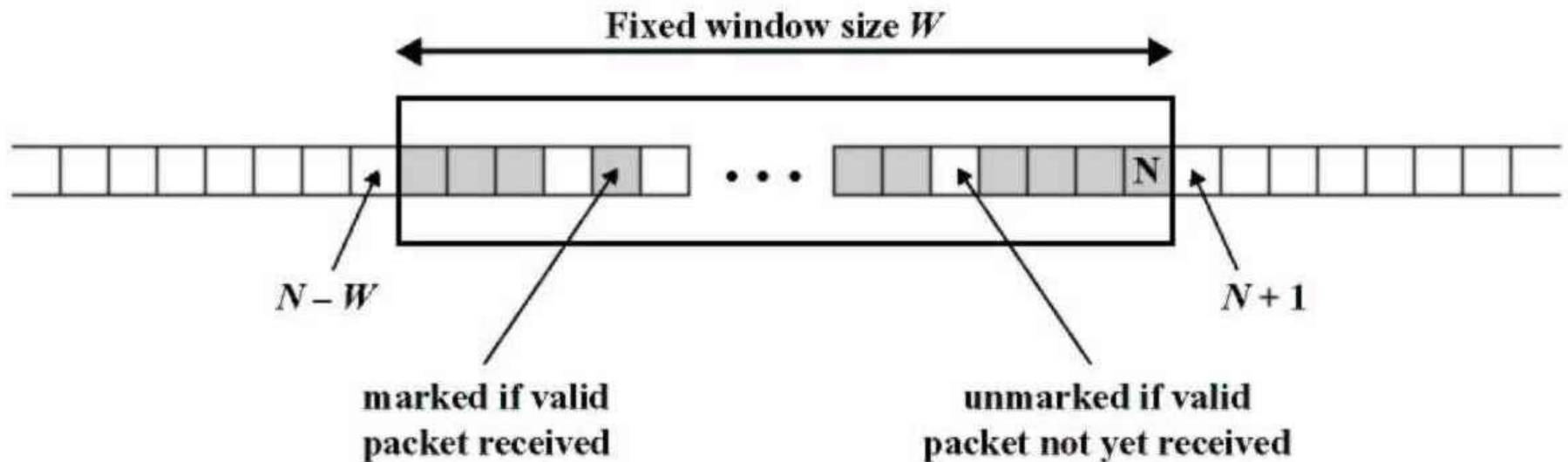
Prevenzione dei replay attack

1. Il ricevente accetta una “finestra” di dimensione W (tipicamente $W=64$) di numeri di sequenza. N sia il massimo numero nella finestra



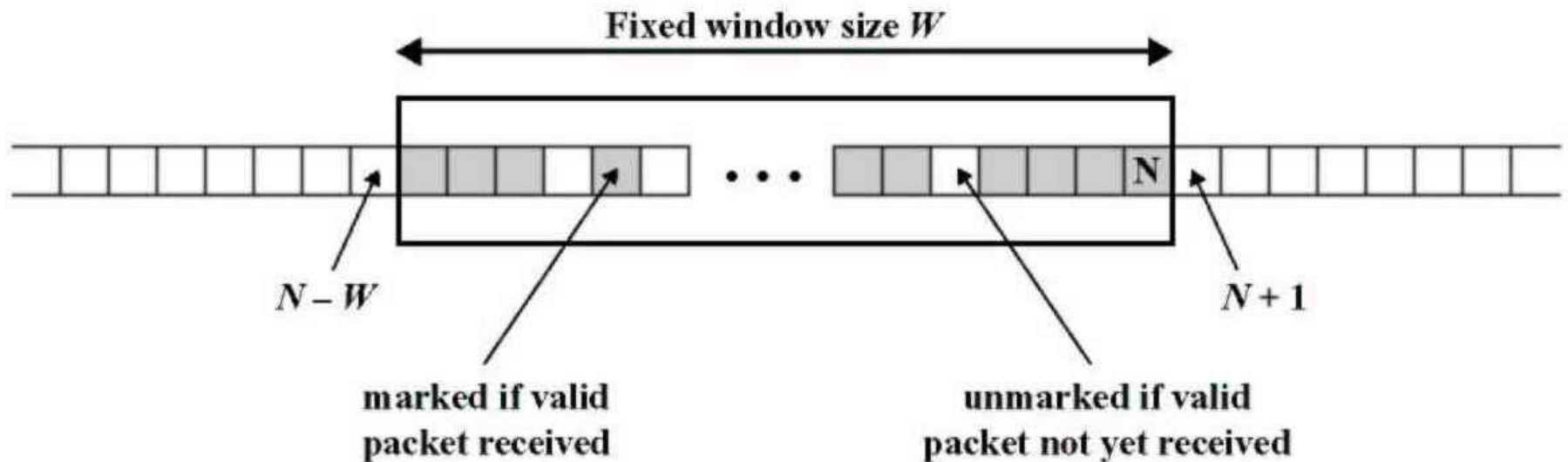
Prevenzione dei replay attack

2. Se il numero di un pacchetto in arrivo **rientra nella finestra**, il pacchetto **non è presente** ed è **autenticato** mediante MAC, viene marcata la posizione relativa



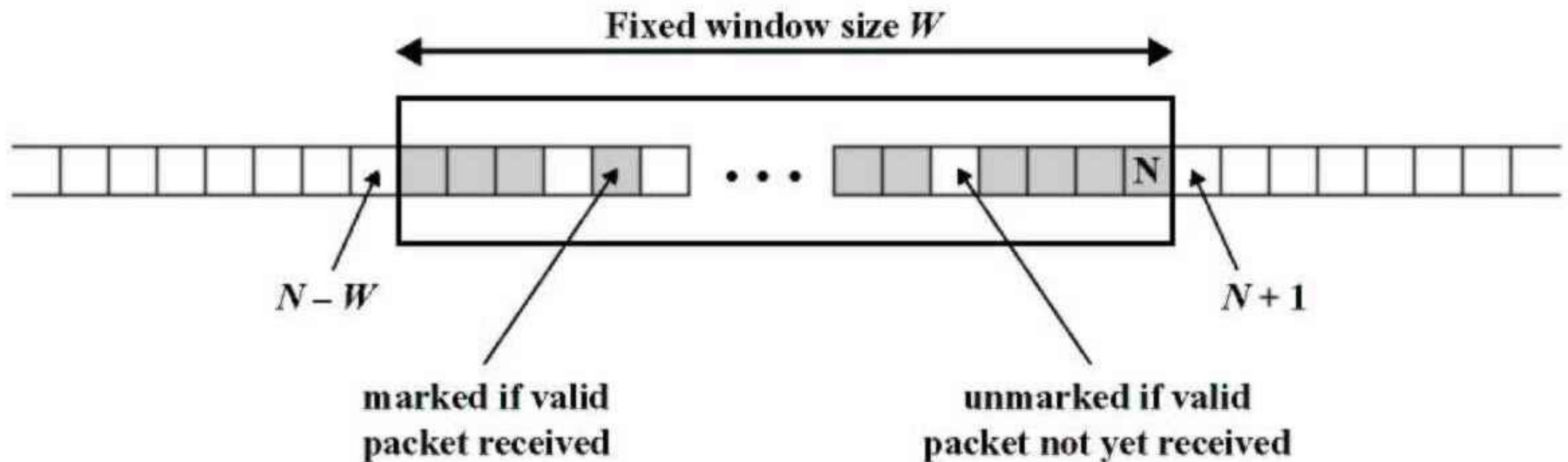
Prevenzione dei replay attack

3. Se il numero di un pacchetto in arrivo è $M > N$, il pacchetto non è presente ed è autenticato mediante MAC, viene estesa la finestra dalla destra fino a M



Prevenzione dei replay attack

4. Se il numero di un pacchetto in arrivo è $M \leq N - W$, oppure il pacchetto **è già presente**, oppure **non è autenticato** dalla MAC, viene segnalata un'anomalia



Protezione mediante AH: trasporto versus tunnel

	Modalità trasporto	Modalità tunnel
Cosa AH protegge mediante il MAC	Dati del pacchetto e campi non variabili dell'header IP (no IP spoofing). In IPv6 anche estensioni dell'header	Intero pacchetto IP (dati + header IP) e campi non variabili dell'IP esterno (no IP spoofing). In IPv6 anche estensioni dell'header

AH in IPv4

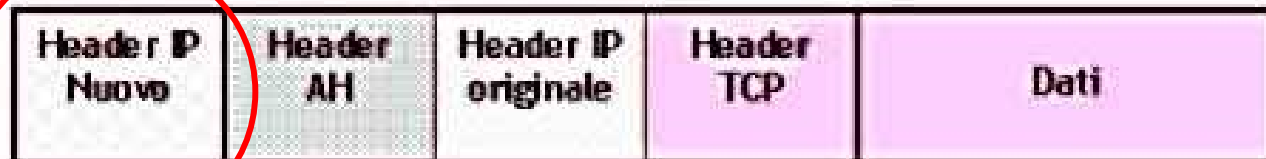


Pacchetto IP originale



modalità
trasporto

Autenticato ad eccezione dei campi variabili e non prevedibili



modalità
tunnel

Autenticato ad eccezione dei campi variabili e non prevedibili

AH in IPv6



← Pacchetto IP originale →



← Autenticato ad eccezione dei campi variabili e non prevedibili →

modalità
trasporto



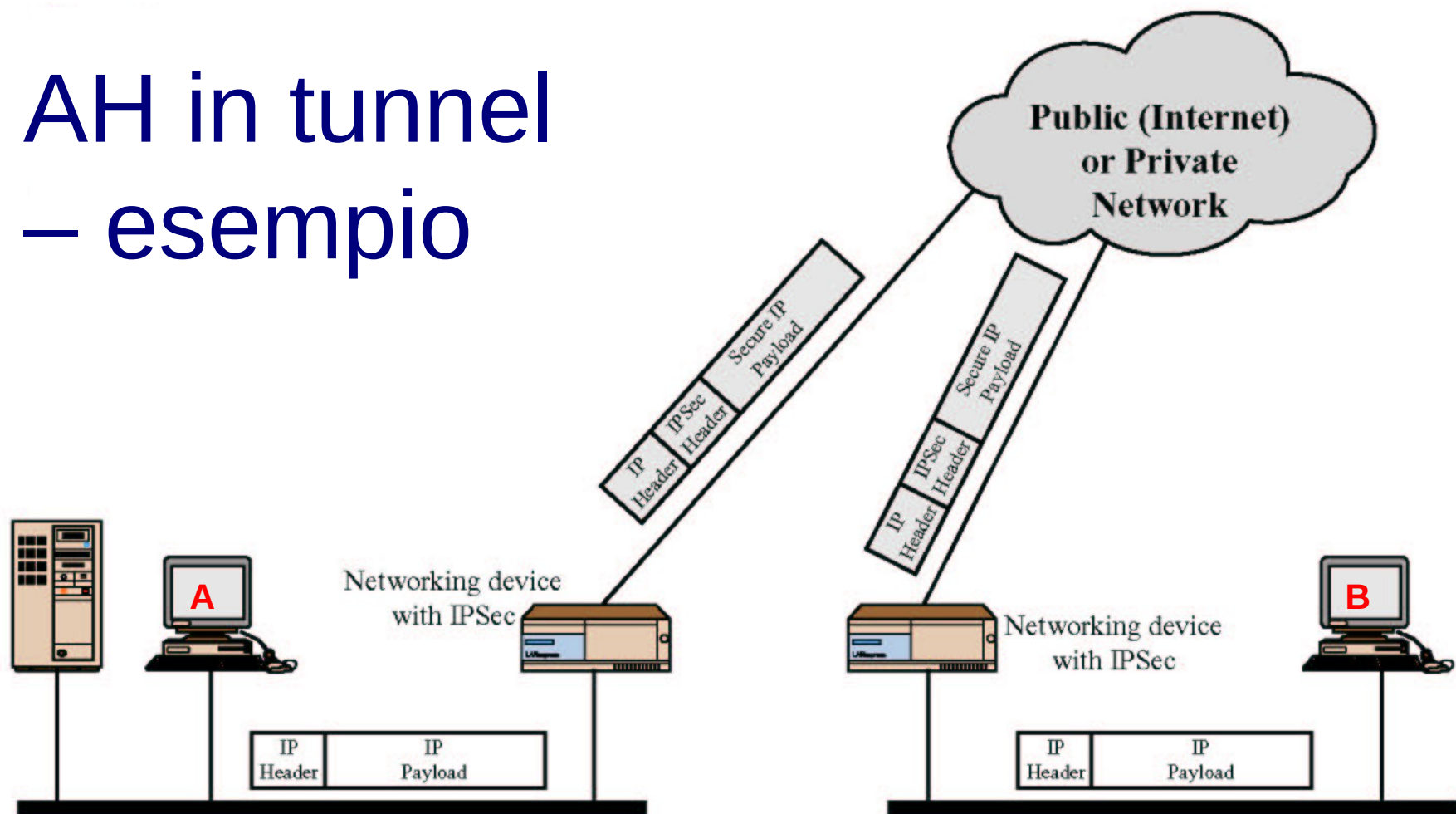
← Autenticato ad eccezione dei campi variabili e non prevedibili del nuovo header IPv6 →

modalità
tunnel

Usi di AH: trasporto versus tunnel

	Modalità trasporto	Modalità tunnel
Requisiti	Mittente e ricevente devono usare IPSec	Mittente e ricevente possono non usare IPSec
Garanzie	Autenticazione punto-punto (end-to-end)	Autenticazione intermedia (attraverso router/firewall)

AH in tunnel – esempio



A e B si scambiano pacchetti autenticati.
I router/firewall intermedi incapsulano il traffico usando AH in modalità tunnel.

AH in tunnel – esempio

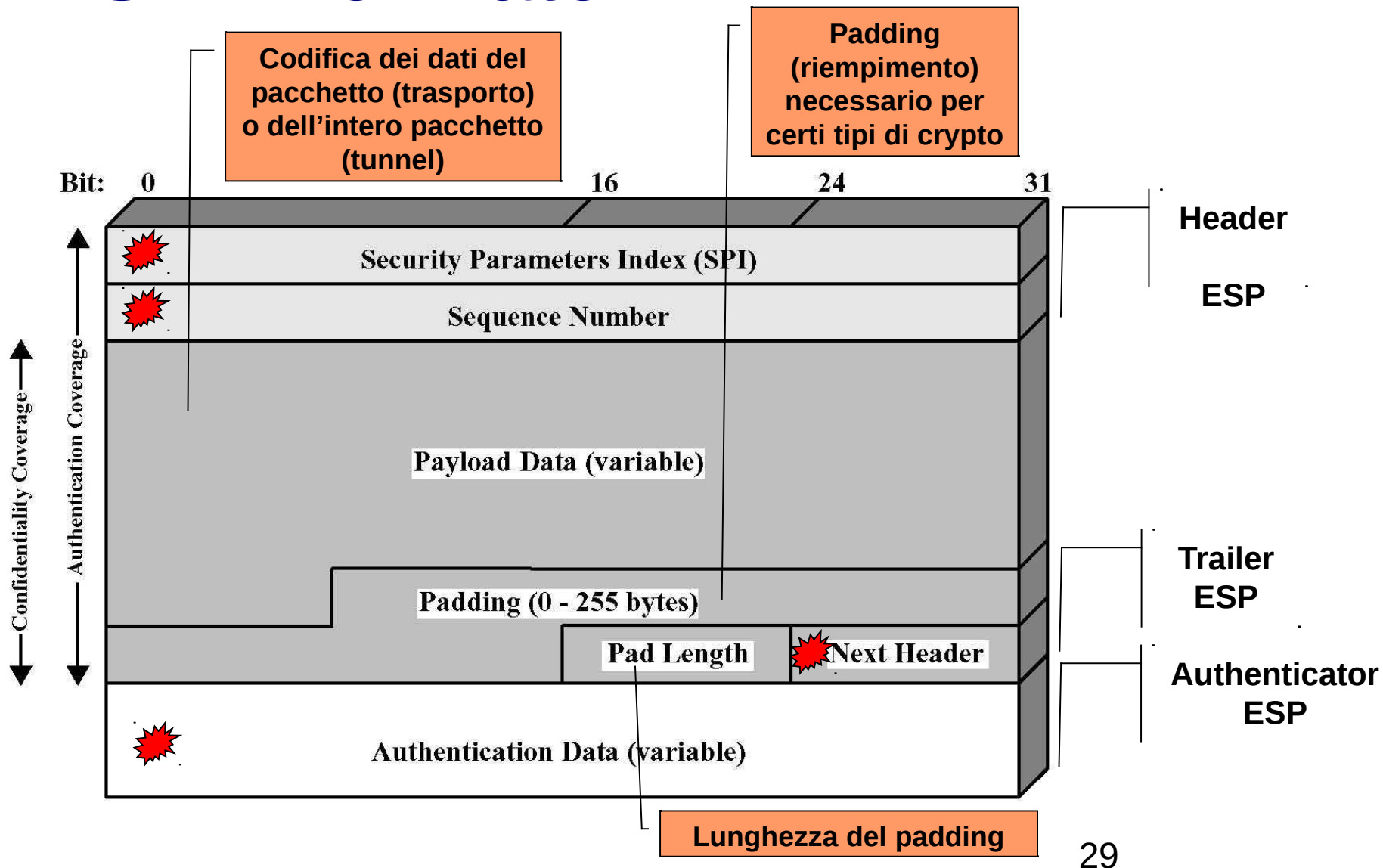
1. Nodo A su rete NA genera pacchetto IP classico per nodo B su rete NB
2. Il router/firewall di NA lo incapsula in un pacchetto IPSec modalità tunnel e lo inoltra al router/firewall di NB
3. Questo estrae ed **autentica** il pacchetto IP originale, poi lo inoltra al nodo B su NB

Encapsulating Security Payload (ESP)

- Fornisce segretezza
 - Del contenuto
 - Del flusso di traffico (contro analisi del traffico, vedi lucido 3.11) mediante uso di intermediari
- Opzionalmente può fornire autenticazione
- Da usare da solo o sequenzialmente insieme ad AH

ESP – formato

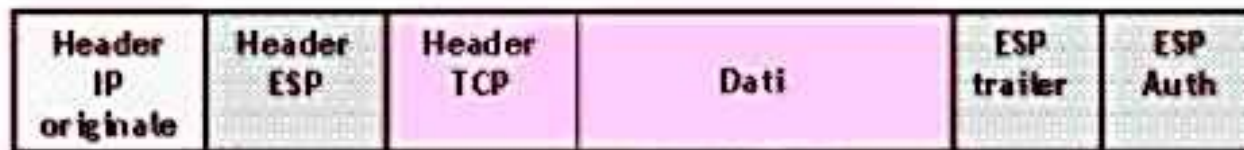
★ = come in AH



ESP in IPv4



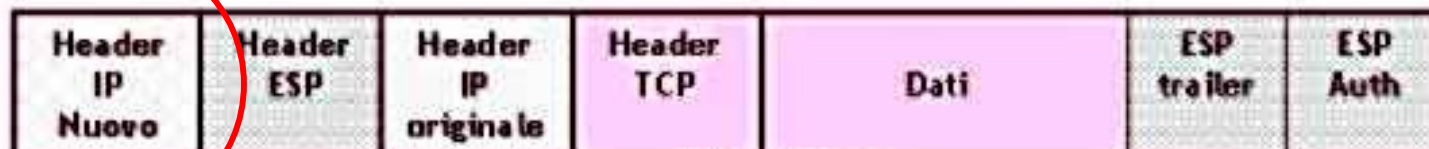
Pacchetto IP originale



Parte cifrata

Parte autenticata

modalità
trasporto

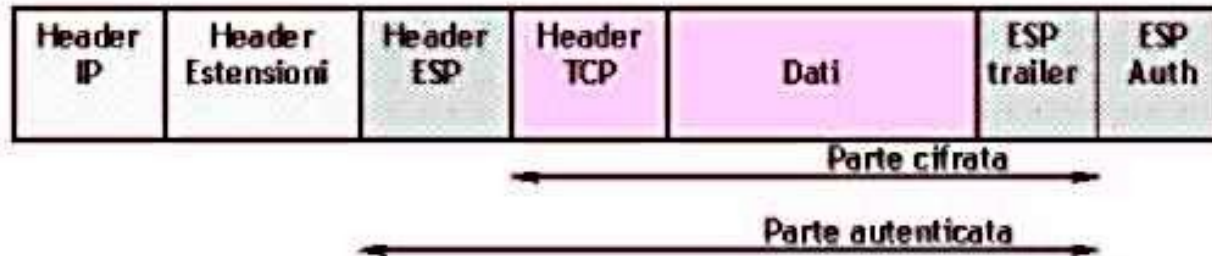


Parte cifrata

Parte autenticata

modalità
tunnel

ESP in IPv6

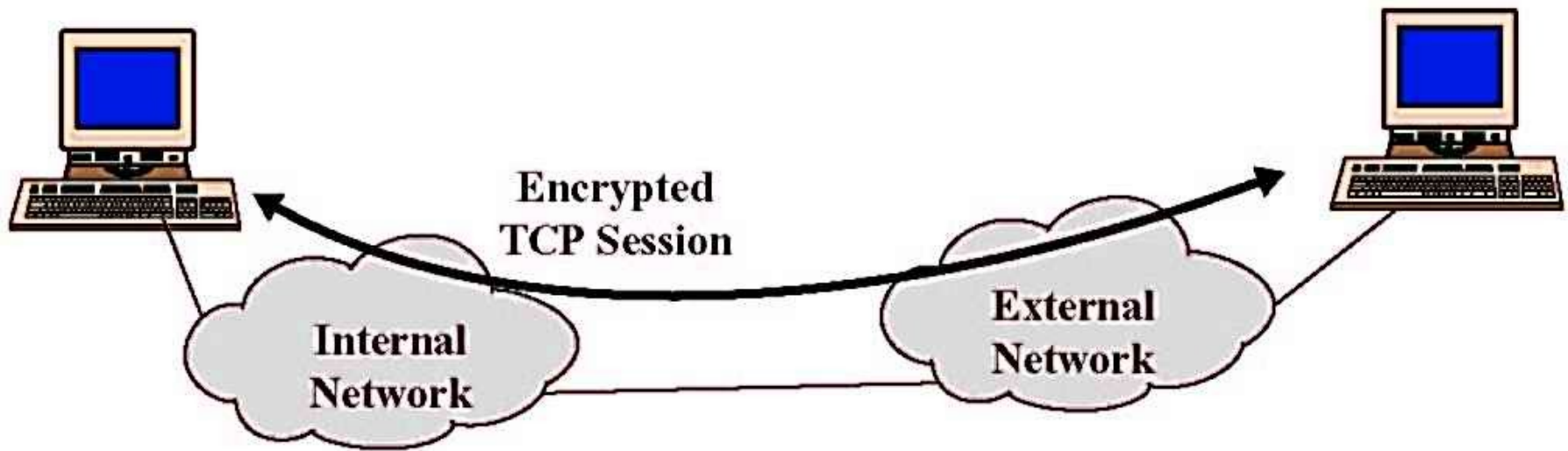


modalità
trasporto



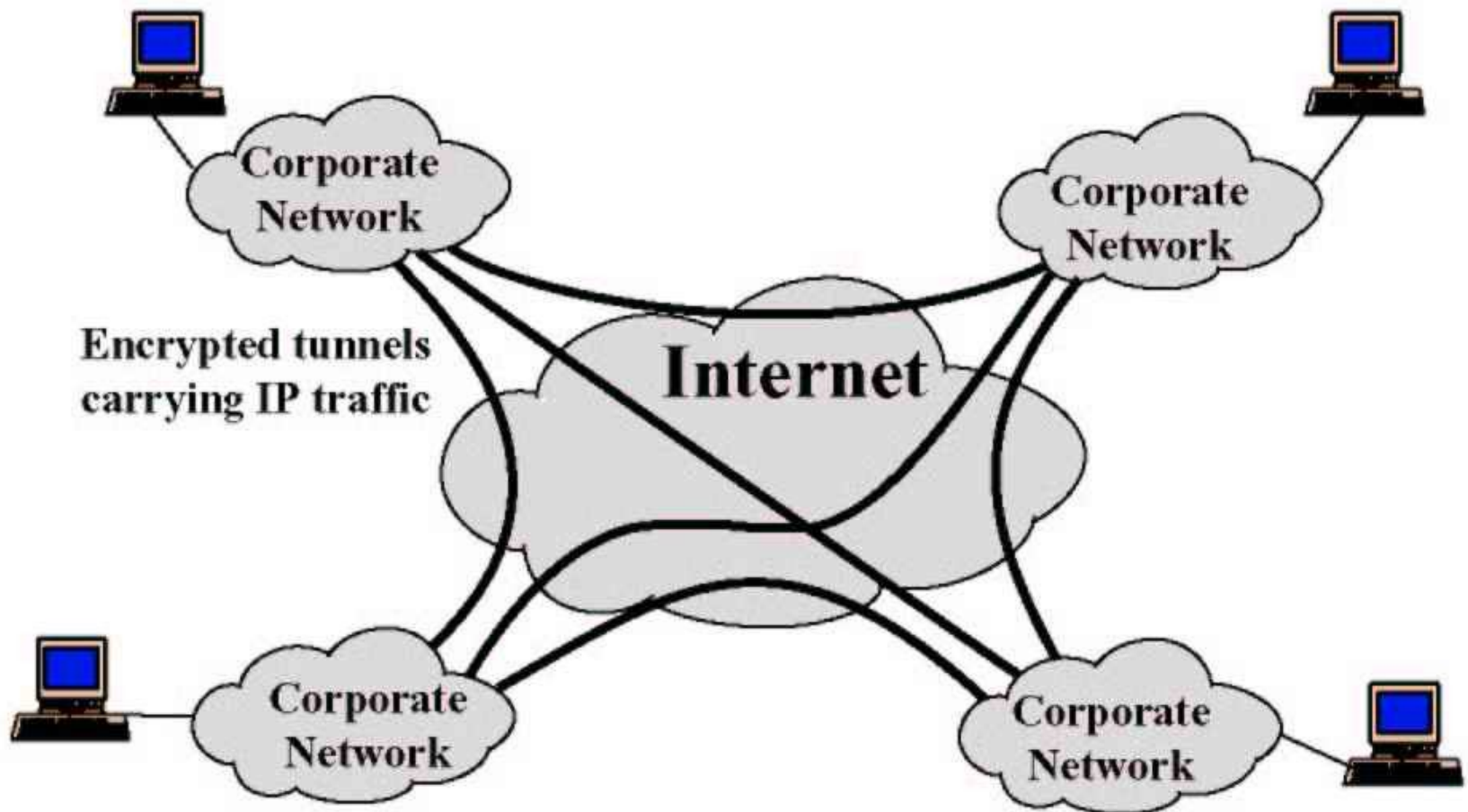
modalità
tunnel

ESP in modalità trasporto



VPN punto-punto – IPSec sui due punti

ESP in modalità tunnel



VPN fra i router/firewall (intermediari) delle varie reti aziendali

ESP in tunnel - esempio

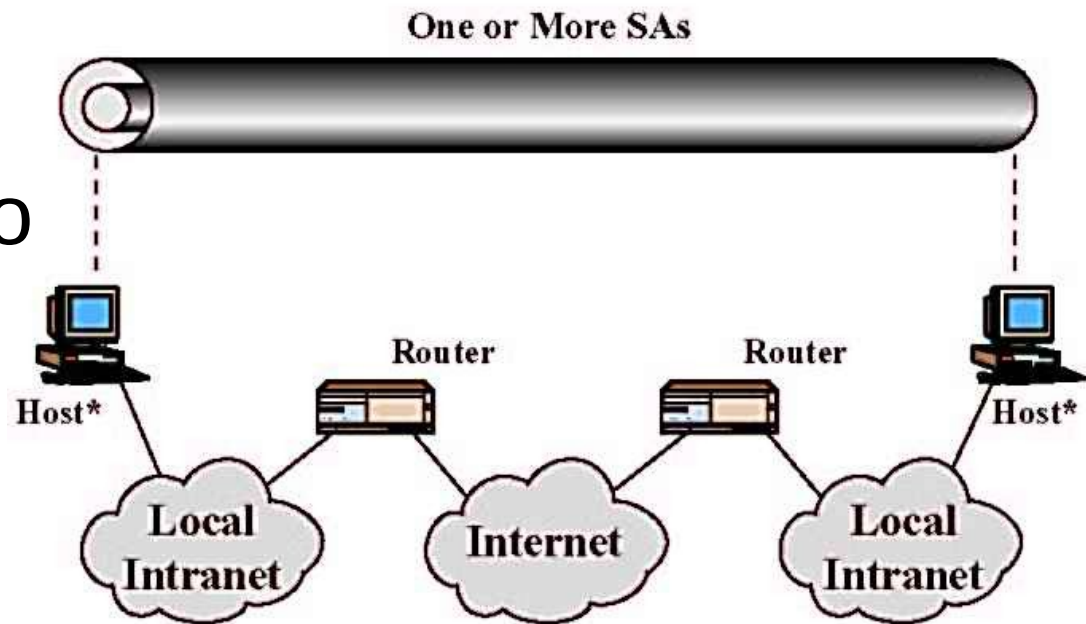
1. Nodo A su rete NA genera pacchetto IP classico per nodo B su rete NB
2. Il router/firewall di NA lo incapsula in un pacchetto IPSec modalità tunnel e lo inoltra al router/firewall di NB
3. Questo estrae, **decripta ed eventualm. autentica** il pacchetto IP originale, poi lo inoltra al nodo B su NB

Combinazioni di SA

- Non ha senso combinare **più di due** modalità di trasporto
 - ESP in trasporto & AH in trasporto
- Può servire combinare (sovrapporre) molteplici tunnel
 - Ogni tunnel può avere propri nodi di inizio e fine
- Ogni nodo compatibile IPSec deve supportare 4 tipi di combinazioni SA

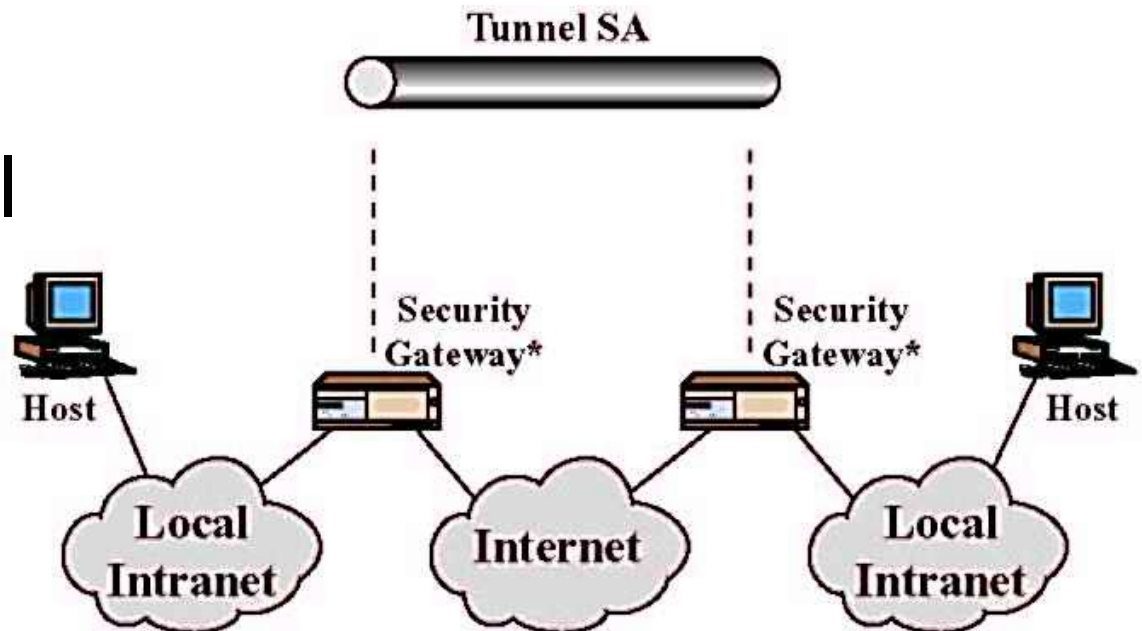
Tipo 1 di combinazioni SA: sicurezza punto-punto

1. AH in trasporto
2. ESP in trasporto
3. 1 & 2



Tipo 2 di combinazioni SA: sicurezza fra intermediari

1. AH in tunnel
2. ESP in tunnel

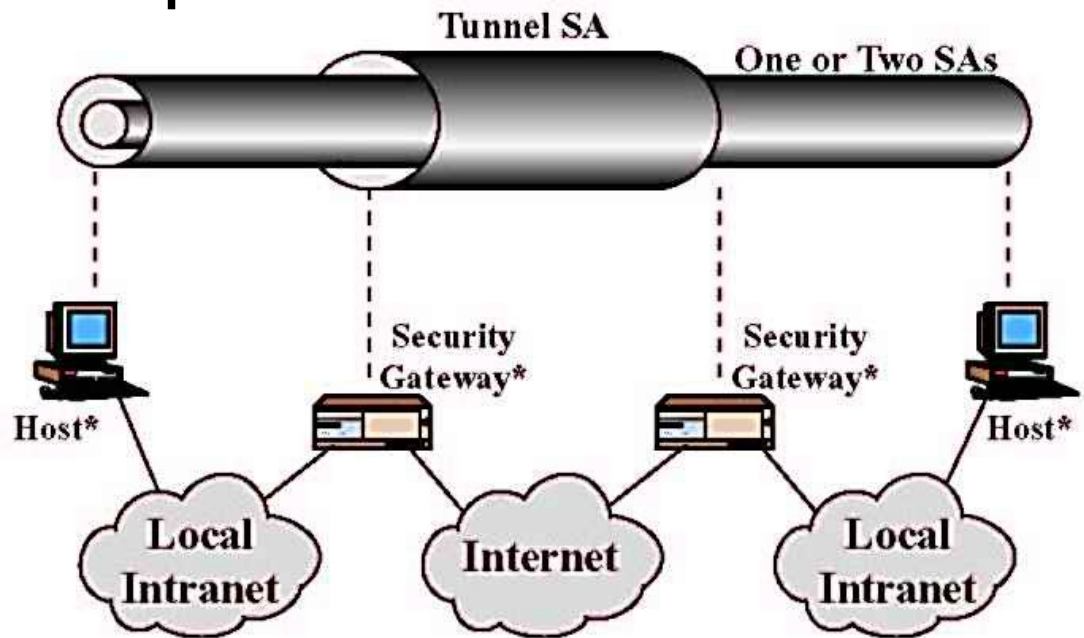


Tipo 3 di combinazioni SA: tipo 1 & tipo 2

1. Combinazione di tipo 1 punto-punto

&

combinazione
di tipo 2 fra
intermediari

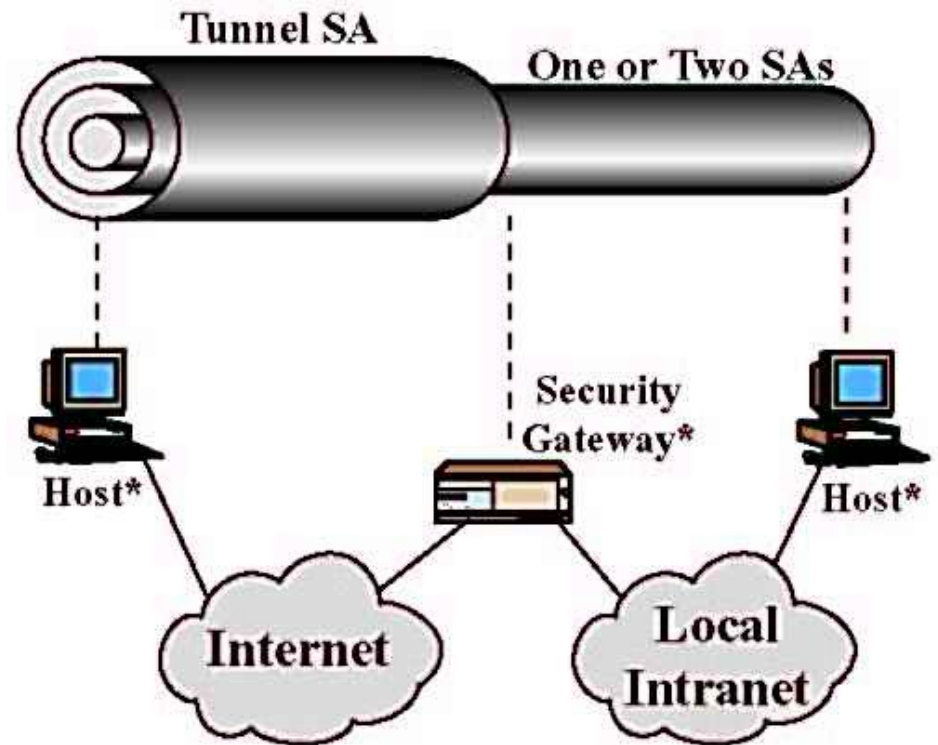


Tipo 4 di combinazioni SA: tipo 1 & sicurezza punto-intermed.

1. Combinazione di
tipo 1 punto-punto

&

(AH in tunnel
punto-intermed. |
ESP in tunnel punto-intermed.)



ESP con/senza autentica

- Una SA (campo 3) può sancire l'utilizzo di ESP con autenticazione
- In questo caso l'uso di AH appare ridondante

Molte opzioni di IPSec appaiono ridondanti

Potenziali semplificazioni di IPSec

1. Eliminare AH: usare sempre ESP con autenticazione
2. Eliminare modalità trasporto e usare sempre solo tunnel
 - ☐ IPSec dovrebbe essere su ogni nodo
 - ☐ Un mondo di VPN
3. Ridefinire le SA come associazioni bidirezionali

Internet Key Exchange (**IKE**)

- Protocollo per la generazione su IP classico delle chiavi da usare poi in AH e/o ESP
- Consta di due protocolli, integrati a livello applicazione
 - **Oakley** (variante di Diffie-Hellmann, livello applicazione), fornisce chiave iniziale
 - **ISAKMP** (Internet Security Association and Key Management Protocol, livello trasporto) crea chiavi di sessione usando la chiave iniziale



Parte 8

Intrusion Detection



Intrusion Detection Working Group

Gruppo di lavoro per lo sviluppo di sistemi (distribuiti) per il rilevamento delle intrusioni. Intende produrre:

1. Una descrizione dei requisiti funzionali di alto livello per le comunicazioni fra i moduli del sistema
2. Un linguaggio per la specifica di intrusioni
3. Una comparativa dei protocolli di comunicazione esistenti più appropriati secondo i requisiti di cui al punto 1

Intrusione versus Malware

- Un intruso è un vero e proprio processo utente (con certi privilegi)
- Raramente i virus possono essere considerati tipi particolari di intruso
 - Un virus non è un batterio
- Più facilmente un worm è un tipo di intruso
 - Un worm è autonomo, come un batterio
- Un intruso **non è** né un virus né un worm ma può **installarne** uno

Intrusione versus DoS

DoS	Intrusioni
<ul style="list-style-type: none">■ Effetti temporanei■ Immediatamente pubblico■ Blocco delle risorse	<ul style="list-style-type: none">■ Effetti generalm. permanenti■ Spesso non reso pubblico■ Uso illecito di risorse

Negazione del servizio (DoS)

Def. Attacco che impedisce la normale operatività di un sistema, degradandola o bloccandola del tutto

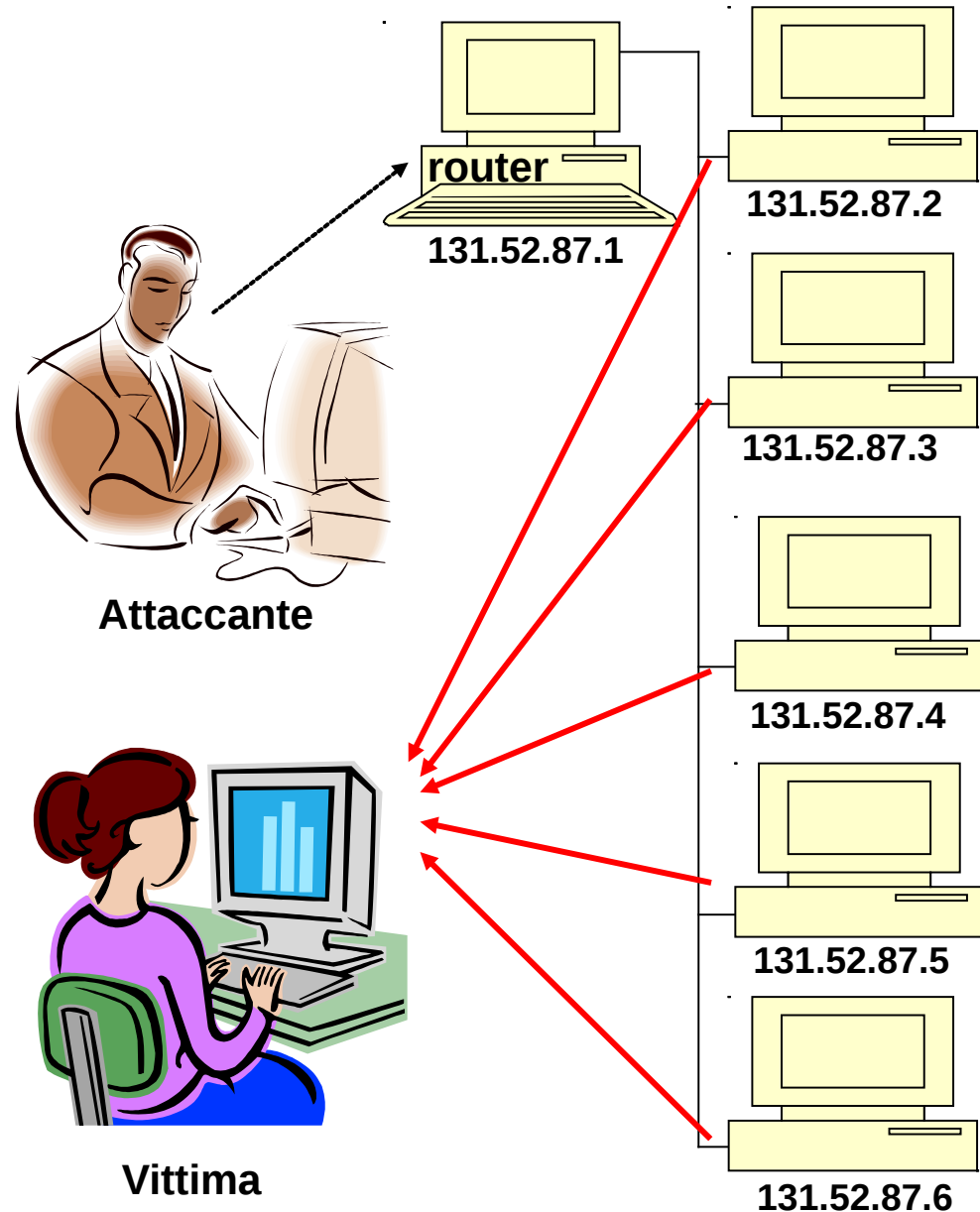
- Tre tipologie
 - **cDoS**: consumo di risorse computazionali
 - **mDoS**: consumo della memoria
 - **bDoS**: consumo della banda di trasmissione
- **DDoS**: DoS sferrato da più macchine
- Indipendenti dall'autenticazione
 - Eccesso di traffico, anche autenticato

Breve storia

- 1998: primi DDoS
- Febbraio 2000: DDoS contro Yahoo!, Amazon, CNN, eBay. Danni circa \$50M
- 2000/2001: moltissimi DDoS a siti aziendali e governativi USA, e ad ISP europei (a partire da Tiscali UK)
- **21 Ottobre 2002**: attaccati i 13 Root DNS di Internet! Conseguenze minime ma vulnerabilità della rete mondiale appurata

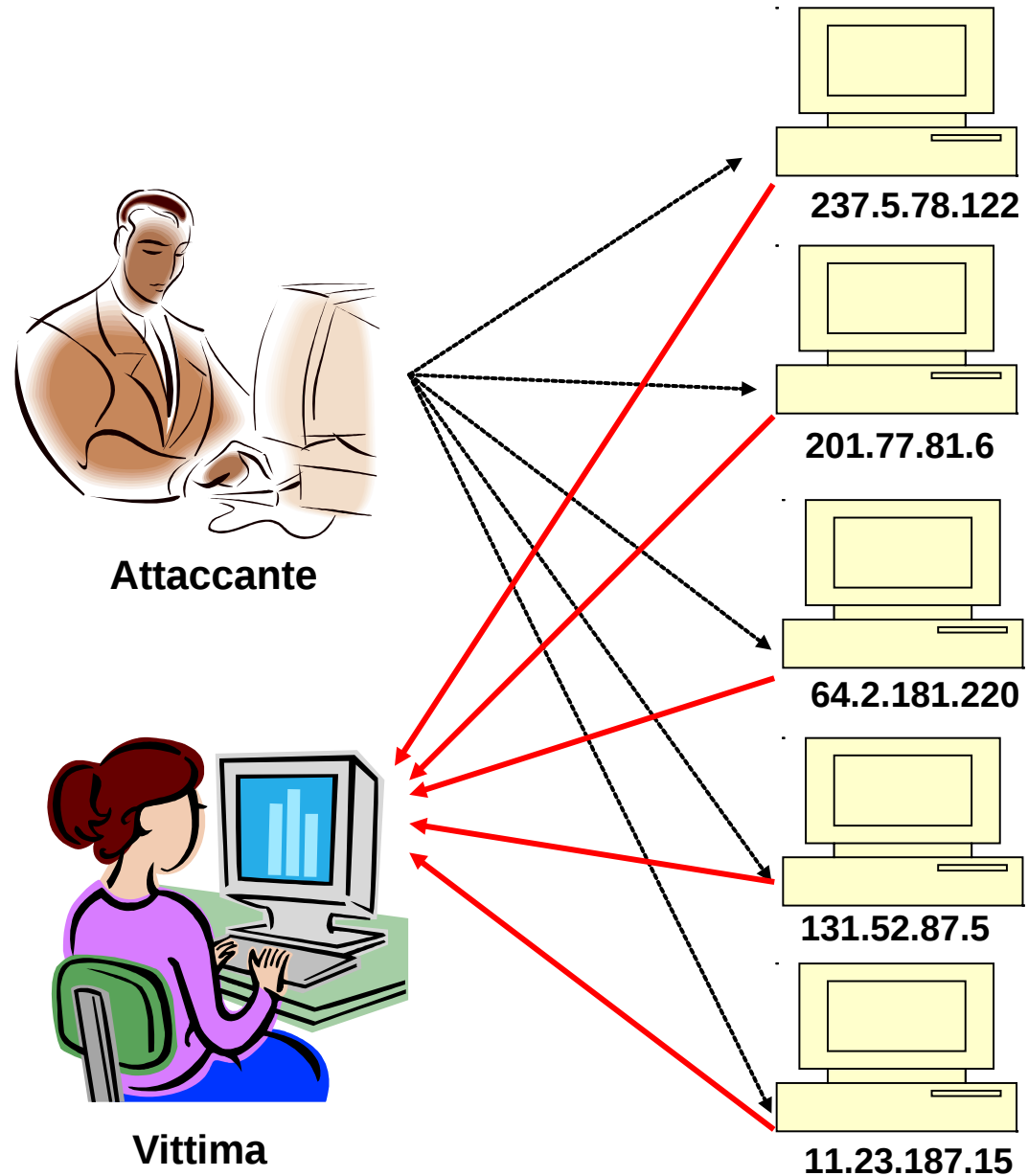
DoS

- Esista un router con servizio di broadcast
- L'attaccante invia un PING utilizzando l'indirizzo della vittima
- Tutte le macchine della sottorete rimbalzano il PING alla vittima



DDoS

- L'attaccante istruisce un numero di zombie di contattare un preciso IP
- La macchina dietro l'IP viene saturata





Esempi

- Ping of death
- Teardrop
- Land
- ...

Semplici da realizzare ma anche da rilevare.

Effetti di un (D)DoS

- Non profitto diretto ma indiretto
 - Mancanza di profitto
 - Perdita di reputazione
- Sempre più ditte basano operatività e presenza sul mercato su sistemi informativi
- Dati USA 2000

Settore	Costo orario (\$)
Energia	2817
Telecomunicazioni	2066
Manifatturiero	1610
Finanza	1495
IT	1344
Assicurazioni	1202
Commercio	1107
Farmaceutica	1082
Bancario	996
Alimentari/bevande	804
Chimica	704
Trasporti	668
Sanitario	636
Elettronica	477
Informazione	340
Alberghiero	330
Valore medio	1010

Come reagire?

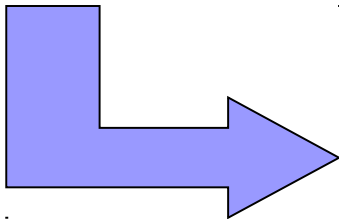
- DoS: chiudere connessioni provenienti dalla singola rete attaccante
- DDoS: chiudere tutte le connessioni?
 - Difficile e sconveniente
- Filtrare il numero di connessioni accettate (htaccess con firewall)
- Un'euristica di prevenzione consiste nel complicare computazionalmente l'accesso (**cookie transformation**)

Anti-DoS: cookie transformation

1. $C \rightarrow S : m1$

2. $S \rightarrow C : m2$

3. $C \rightarrow S : m3$



1. $C \rightarrow S : m1$

2. $S \rightarrow C : m2_s$

3. $C \rightarrow S : m1, m2_s, h(m1, m2_s)$

4. $S \rightarrow C : m2_c$

Creazione del messaggio computazionalmente complicato rimandata fino a quando è chiaro che il client sia pronto a ricevere all'IP del passo 1

Tool per DDoS

Nome tool	Tecniche
Trin00	UDP
TFN	UDP, ICMP, Smurf
TFN2K	UDP, ICMP, Smurf
Stacheldracht	UDP, ICMP, ACK, Smurf
Shaft	UDP, ICMP, Smurf
Mstream	ACK
Trinity	UDP, ICMP, ACK, flags

Tool per DDoS – elementi

■ Funzionamento

- Saturazione banda
- Pacchetti malformati: *Ping of Death, Teardrop, Land, Frammenti*

■ Connessioni

- TCP e UDP: *Trin00, TFN*
- Connessioni criptate: *TFN2K, Stacheldracht*
- Connessioni via IRC (chat): *Trinity*

Tool per DDoS – elementi

- Metodi di Spoofing dell'indirizzo iniziale
 - Random sull'intero indirizzo IP
 - Random sull'ultimo byte
- Piattaforme
 - Da TFN2K: sia Unix che Windows
 - Risolte alcune non indifferenti problematiche di interconnessione
 - *Ubiquitous computing*

Tool per DDoS – elementi

- Distribuzione frammenti codice
 - Uso di vulnerabilità note ed automatizzabili
 - Allegati di posta elettronica
 - Worm
- Macchine coinvolte
 - DNS, Web Server, utenze personali “always-on”, società dotate di connessioni veloci, siti di e-commerce, università,...

Intrusione versus DoS

DoS	Intrusioni
<ul style="list-style-type: none">■ Effetti temporanei■ Immediatamente pubblico■ Blocco delle risorse	<ul style="list-style-type: none">■ Effetti generalm. permanenti■ Spesso non reso pubblico■ Uso illecito di risorse

Intrusione

Def. Ottenere illecitamente privilegi superiori a quelli posseduti lecitamente

- Acquisire un accesso al sistema oppure ampliare i propri privilegi di accesso
- Non esclusivamente da rete
 - *Inserire SW nocivo mediante floppy o pen-drive*
- Si riduce spesso a violare i meccanismi di autenticazione e/o controllo d'accesso



Tecniche di intrusione

- **Violazioni di password:** tecniche standard e non-standard (vedi lucido 5.*)
- **Intercettazione di informazioni sensibili:** scoparle (non solo password) se transitano in rete senza crittografia oppure violare protocolli di sicurezza
- **Uso di combinazioni di SW nocivo:** sullo stile del worm Morris

Trattamento delle intrusioni

Def.

- Rilevare l'intrusione prima possibile (**intrusion detection**)
- Espellere l'intruso non appena rilevato per minimizzare i danni che può provocare (**intrusion management**)

IDS (Intrusion Detection System)

- Definizione ovvia
- Registra il comportamento del sistema (**auditing**) e analizza tale log
 - Tecniche di pattern matching
 - Funzionamento real-time
- IDS distribuito: più processi o macchine
 - Registrazione
 - Analisi
 - Decisione

Definire i comportamenti

- Il problema fondamentale per un IDS è definire il comportamento dell'intruso rispetto a quello dell'utente legittimo
 - Comportamento medio nel tempo
 - Machine learning
- Trovare miglior compromesso fra scambiare
 - Utente legittimo per intruso (falso positivo)
 - Intruso per utente legittimo (falso negativo)

Record di auditing

1. Record nativi

- Tutti i S.O. prevedono la raccolta di informazioni generali sulle attività degli utenti
 - *Storia dei comandi di shell*
- Semplice, ma non sempre dà informazioni utili

2. Record specifici per il rilevamento

- SW aggiuntivo che raccoglie esclusivamente informazioni specifiche per il rilevamento
 - *Deviazione dal modello statistico*
- Appesantisce il sistema, ma dà info mirate

Record specifici per il rilevamento

- Le operazioni classiche vengono tradotte in una sequenza di azioni elementari
 - *Copiare un file richiede 3 azioni*
 1. *Esecuzione del programma di copia*
 2. *Lettura del file da copiare*
 3. *Scrittura del file da copiare*
- Più semplice controllare i tentativi di aggirare i controlli di accesso

Record di Denning

E' un esempio di record specifici per il rilevamento, ciascuno contenente 6 campi

1. **Soggetto**: utente, o processo, che esegue azioni per mezzo di appositi comandi.
 - Raggruppabili in classi di accesso (ruoli)
2. **Azione**: operazione svolta dal soggetto
3. **Oggetto**: ciò su cui viene svolta l'azione
 - Un soggetto può essere oggetto

Record di Denning

4. **Eccezione**: se e quale eccezione sia stata prodotta in risposta all'azione
5. **Uso risorse**: quali e quanti elementi usati
 - *Numero righe stampate o visualizzate*
 - *Numero di settori letti o scritti*
 - *Tempo CPU impiegato*
 - *Unità di I/O utilizzate*
6. **Timestamp**: identificatore temporale per il momento di inizio dell'azione

Record di Denning – esempio

- Il comando
cp ~giamp/slides.pdf ~barba
- Genera i 3 record relativi alle 3 azioni

giamp	execute	/bin/cp	0	CPU = 00002	11058721678
giamp	read	~giamp/slides.pdf	0	SECTORS = 5	11058721679
giamp	write	~barba	-1	SECTORS = 0	11058721680

anomalia



Tecniche di rilevamento

1. Indipendenti dal passato (standard)
 - ☐ *Max 3 fallimenti di inserimento di password...*
2. Dipendenti dal passato
 - ☐ Osservazione/campionamento del funzionamento/comportamento del sistema in passato
 - ☐ Assunzione che il futuro sarà come il passato

Tecniche di rilevamento

1. Rilevamento statistico

- Creazione di un modello statistico del comportamento di un **utente legittimo**
- L'intruso si discosta sufficientem. dal modello
- Particolarmente efficace per attaccanti esterni

2. Rilevamento a regole

- Creazione di regole per il comportamento di un **intruso**
- Particolarmente efficace per attaccanti interni

Rilevamento statistico – modelli

1. **Media e deviazione standard**: calcolati su un parametro in un arco di tempo, danno l'idea del comportamento medio e della sua variabilità
2. **Modello operativo**: definisce un limite di accettabilità per un parametro e segnala una intrusione quando viene superato
3. **Multivariiazione**
4. **Processo di Markov**
5. **Serie temporale**

Rilevamento statistico – esempi

Parametro	Modello	Tipo di intrusione rilevata
Login and Session Activity		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off-hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.

Rilevamento statistico – esempi

Parametro	Modello	Tipo di intrusione rilevata
Command or Program Execution Activity		
Execution frequency	Mean and standard deviation	May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands.
Program resource utilization	Mean and standard deviation	An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization.
Execution denials	Operational model	May detect penetration attempt by individual user who seeks higher privileges.

Rilevamento statistico – esempi

Parametro	Modello	Tipo di intrusione rilevata
File access activity		
Read, write, create, delete frequency	Mean and standard deviation	Abnormalities for read and write access for individual users may signify masquerading or browsing.
Records read, written	Mean and standard deviation	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.
Failure count for read, write, create, delete	Operational	May detect users who persistently attempt to access unauthorized files.

Rilevamento a regole

- Grosso database di regole, fino a 10^4 - 10^6
 - *Nessun utente dovrebbe leggere i file contenuti nelle directory personali di altri utenti*
 - *Nessun utente dovrebbe scrivere su file di altri utenti*
 - *Gli utenti che si connettono al sistema dopo ore spesso accedono a file utilizzati in precedenza*
 - *Nessun utente generalmente accede direttamente ai dischi bensì usa servizi di alto livello offerti dal S.O.*
 - *Nessun utente dovrebbe trovarsi connesso più volte allo stesso sistema*
 - *Nessun utente esegue copie di eseguibili*
- Es: ntop, snort
 - A regole ma basati su firme degli attacchi

Tecniche di protezione

1. **Limitare l'automazione**: impedire il più possibile il ripetersi di tentativi
 - *Dopo 3 inserimenti di password errata, il sistema resetta la connessione*
 - *Dopo 3 inserimenti di PIN errato, il telefonino blocca la SIM*

Tecniche di protezione

2. **Usare esche**: risorse fittizie che attirano gli attaccanti

- Magari distogliendoli da risorse più importanti
- O convincendoli a rimanere di più nel sistema
 - transazioni.grossasocieta.com
 - conticorrenti.banca.com

3. **Usare trappole**: massiccio monitoraggio appena l'attaccante accede all'esca





IMS (Intrusion Management System)

Def. L'intrusione va

1. Contenuta (containment)
2. Rimossa (eradication)
3. Riassorbita (recovery)
4. Punita (punishment)

Intrusion Containment

- Permettere collegamenti solo su VPN
- Monitorare l'attaccante passivamente
- Limitare le sue attività
 - Diminuirgli i privilegi d'accesso
 - Aumentare la protezione delle risorse
 - *E se stacciamo la spina?!*



Intrusion Eradication

- Chiudere tutte le connessione alla rete
- Terminare i processi dell'attaccante
- Bloccare attacchi futuri
 - Chiudere certe porte
 - Disabilitare specifici indirizzi IP



Intrusion Recovery

- Ovvvia tecnica di recovery
 - Memorizzare periodicamente lo stato del sistema
 - Ripristinare l'ultimo stato (sicuro) memorizzato prima dell'intrusione

Intrusion Punishment

- Azione legale
 - Difficile rintracciare attraverso la rete
- Tagliare le risorse
 - Notificare l'ISP
- Controattacco
 - Da studiare e valutare attentamente



Parte 11

Firewall

Misure di sicurezza (lucidi 2.*)

1. Limitazione dei rischi
2. Uso di deterrenti
3. Prevenzione
 - **Firewall**
4. Rilevamento
5. Reazione

L'uso di firewall non è l'unica né la prima linea di difesa

Usare
con cura



Firewall (muro antifuoco) 1992

Def. Dispositivo che controlla il flusso di traffico fra reti con diverse impostazioni di sicurezza

1. Fra una LAN e la rete esterna

- *DMI è protetto da firewall*

2. Fra sottoreti interne

- *Un firewall regola il flusso fra la VLAN per gli studenti e quella per i docenti*

Requisiti di un firewall

1. Tutto il traffico fra una rete e l'altra deve passare attraverso il firewall
 - Impedire accesso a Internet se non via firewall
2. Il passaggio del traffico è regolato da un'apposita politica di sicurezza
 - *Passano tutti i pacchetti in uscita*
3. Il firewall deve essere sicuro esso stesso
 - Installato su un **bastion host**

Bastion host

- Macchina su cui gira un firewall, punto nevralgico per la sicurezza del sistema
- Irrobustita con numerose accortezze
 1. Sono installati solo servizi essenziali, quali Telnet, DNS, FTP, SMTP
 2. Sono servizi semplificati (proxy)
 3. Ciascun servizio richiede propria autenticaz.
 4. Non fanno accessi al disco se non per leggere file iniziali di configurazione
 5. Ogni servizio gira come utente standard



Funzionalità di un firewall

1. Proteggere le risorse interne

- *Dettagli sul file system, sugli account, ...*

2. Monitorare il traffico

- Fondamentale insieme a tecniche di gestione delle intrusioni

3. Filtrare i dati

- *Allegati email, applet Java, controlli ActiveX*

Funzionalità di un firewall

4. Creare VPN

- *IPSec modalità tunnel sul firewall*

5. Mappare indirizzi locali in indirizzi Internet (**NAT** – Network Address Translator)

- Utile contro IP spoofing

6. Fornire IP dinamici (**DHCP** – Dynamic Host Configuration Protocol)

- Semplifica gestione dell'indirizzario IP



Politiche di sicurezza adottate

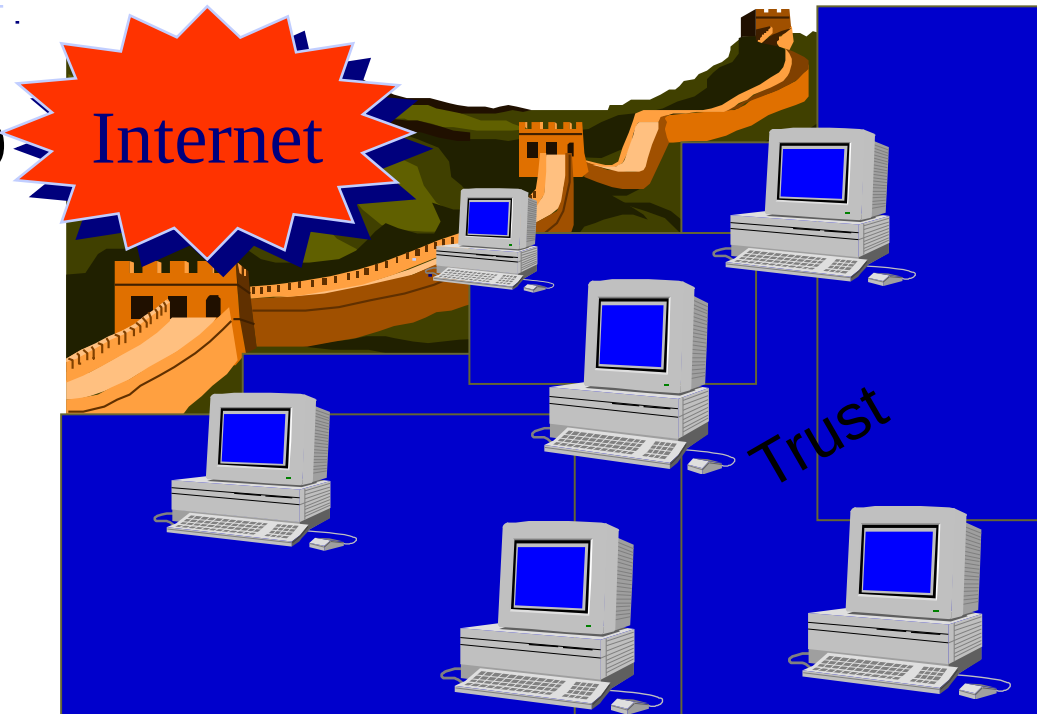
1. **Default deny**: tutto quello che non è espressamente permesso è vietato
2. **Default permit**: tutto quello che non è espressamente vietato è permesso

Non c'è mai discrezionalità: non c'è mai il non obbligo di fare

La difesa perimetrale

1. Mantenere un'unica porta di accesso ad Internet crea delle relazioni di fiducia fra le macchine che si trovano all'interno della rete

- ☐ Pro
- ☐ Contro



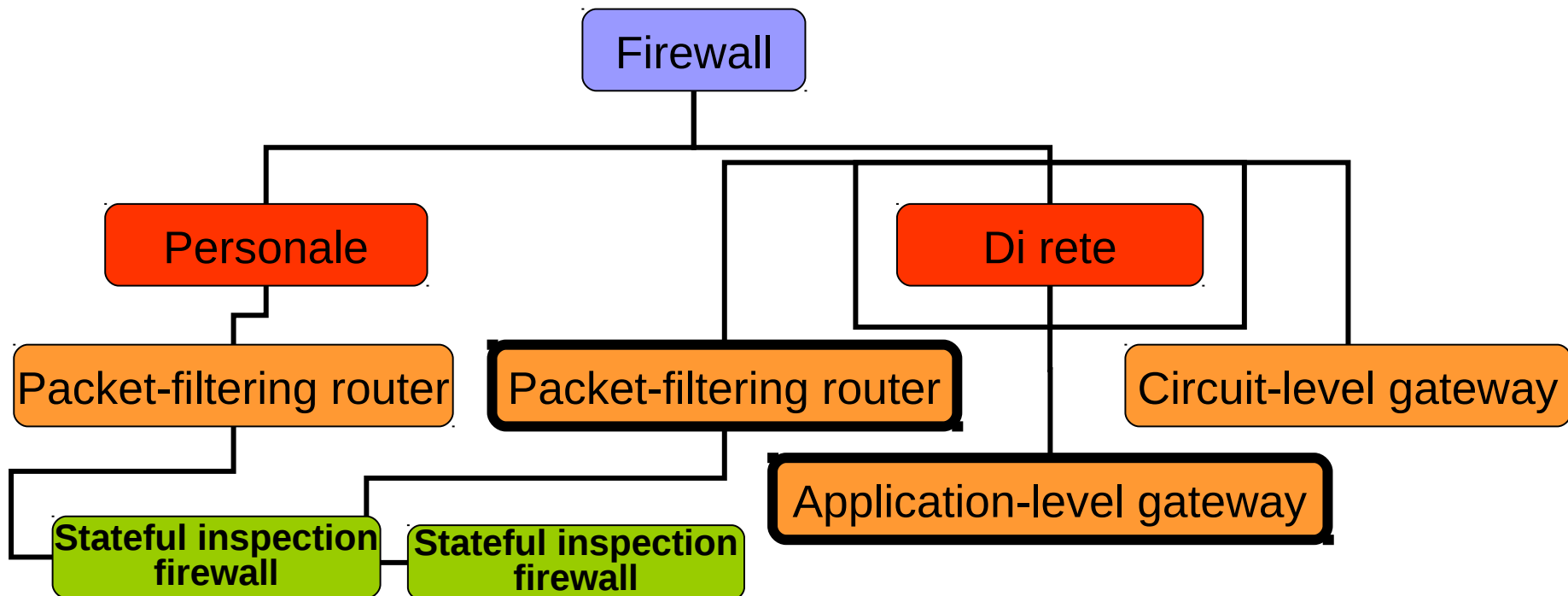
Limiti di un firewall

1. Non possono proteggere da attacchi che ricevono il permesso di superare il firewall
 - *Connessioni via modem*
2. Non possono proteggere da minacce interne
 - *Un utente interno potrebbe essere “bad”*
3. Proteggono minimamente dal passaggio di software nocivo
 - Setacciare tutto il traffico sarebbe impraticabile

Limiti di un firewall

4. Possono degradare le prestazioni della rete
 - Filtraggio e monitoraggio
5. Possono essere difficili da configurare
 - Difficile compromesso fra libertà e sicurezza
6. Non possono proteggere da attacchi non ancora documentati ai protocolli di sicurezza
 - *Siamo dietro il firewall ma hanno scoperto i dettagli della mia transazione*

Firewall – tassonomia

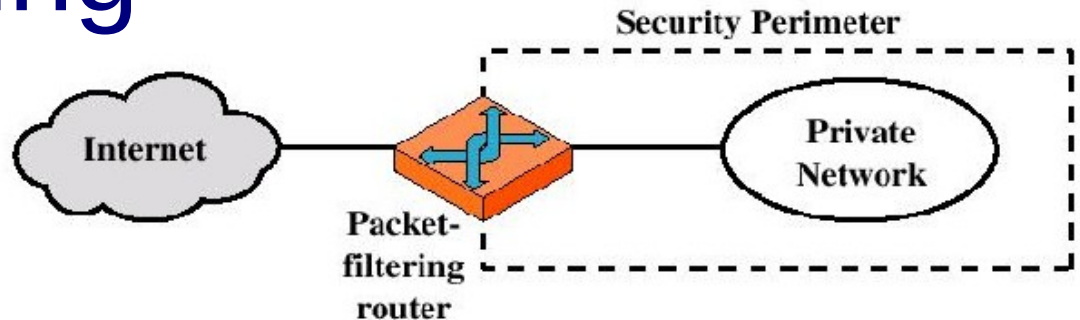




Firewall personale

- Protegge solo la macchina su cui gira
- Tuttora non molto diffuso
- Particolarmente opportuno per utenti mobili
- Utilizzabile insieme a firewall di rete
- Se una ditta impone una politica molto rigida sui propri laptop, questi difficilmente possono essere usati anche per scopi non lavorativi
 - Servirebbero macchine separate

Packet-filtering router



- Livello rete
- Applica un insieme di regole ad ogni pacchetto in ingresso
- Al meglio anche ai pacchetti in uscita
- Le regole sono basate su informazioni contenute nel pacchetto
- Vengono applicate con priorità decrescente



Regole per packet-filtering router

1. **Indirizzo IP di origine**
2. **Indirizzo IP di destinazione**
3. **Indirizzi di origine e destinazione a livello trasporto**: numero porta a livello trasporto (TCP o UDP) che definisce i servizi
4. **Protocollo IP**: definisce il protocollo
5. **Interfaccia**: per un router con 3 o più porte, definisce l'interfaccia di provenienza e destinazione

Regole bidirezionali – esempio 1

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

1. Tutto il traffico proveniente da SPIGOT viene bloccato
2. Posta in ingresso permessa solo sul nostro gateway
3. Default deny
4. Priorità decrescente

Regole bidirezionali – esempio 2

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

1. Aggiunta alla fine di una qualunque lista di regole, realizza la politica default deny

Regole bidirezionali – esempio 3

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

1. Concede alle macchine interne di mandare posta alle esterne
2. Porta 25 collegata al servizio di posta per default
3. Un attaccante esterno potrebbe collegare un altro servizio alla 25 e spedire pacchetti che attraverserebbero il firewall

Regole unidirezionali – esempio 1

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

1. Una lista di macchine interne può mandare pacchetti di posta all'esterno
2. Riceviamo solo pacchetti di posta contenente il flag ACK, ossia di replica

Regole unidirezionali – esempio 2

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

1. I pacchetti di una lista di macchine interne possono uscire
2. Qualunque pacchetto di replica (ACK) può entrare
3. Qualunque pacchetto destinato a una porta alta può uscire

Regole unidirezionali – esempio 2

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

1. E' un modo per gestire un trasferimento FTP
 - Prima connessione, di controllo, alle porte basse
 - Seconda, per trasferimento dati, alle porte alte
2. Il servizio FTP va configurato coerentemente con le regole sia su src che su dest
3. Difficile gestire applicazioni a questo livello

Un insieme realistico

	Source Address	Source Port	Destination Address	Destination Port	Action	Description
1	Any	Any	192.168.1.0	> 1023	Allow	Rule to allow return TCP Connections to internal subnet
2	192.168.1.1	Any	Any	Any	Deny	Prevent Firewall system itself from directly connecting to anything
3	Any	Any	192.168.1.1	Any	Deny	Prevent External users from directly accessing the Firewall system.
4	192.168.1.0	Any	Any	Any	Allow	Internal Users can access External servers
5	Any	Any	192.168.1.2	SMTP	Allow	Allow External Users to send email in
6	Any	Any	192.168.1.3	HTTP	Allow	Allow External Users to access WWW server
7	Any	Any	Any	Any	Deny	"Catch-All" Rule - Everything not previously allowed is explicitly denied



Packet-filtering router – pro

1. Struttura semplice
2. Trasparenza per l'utente
3. Prestazioni



Packet-filtering router – contro

1. Mancanza di informazioni di più alto livello nello stack TCP/IP
 - Le applicazioni possono solo essere consentite o meno, le loro funzionalità non possono essere gestite
 - Funzione di monitoraggio inefficace, raccoglie solo gli elementi visti nelle regole
 - Non supporta autenticazione utente

Packet-filtering router – contro

2. IP spoofing

- L'attaccante manda pacchetti dall'esterno utilizzando IP interni
- Spera che ci siano regole che accettano tali pacchetti
- Soluzione: regole appropriate e/o NAT



Packet-filtering router – contro

3. Attacchi di instradamento (routing)

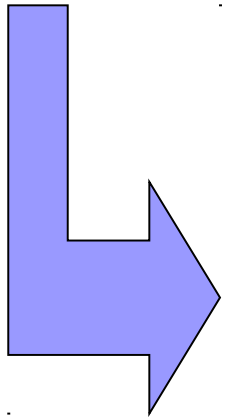
- L'attaccante imposta routing stretto attraverso la rete protetta da firewall
- I pacchetti potrebbero entrare se ci fossero regole permissive per il nodo immediatamente precedente nel percorso
- Soluzione: regole opportune per l'origine dell'instradamento

Stateful inspection firewall

- Variante di packet-filtering router
- Sfrutta informazioni provenienti dal livello di trasporto per gestire meglio le applicazioni
- Concede ingresso dati solo sulla specifica porta richiesta
 - Evita gli enormi rischi derivanti dall'aprire tutte le porte alte (> 1023)

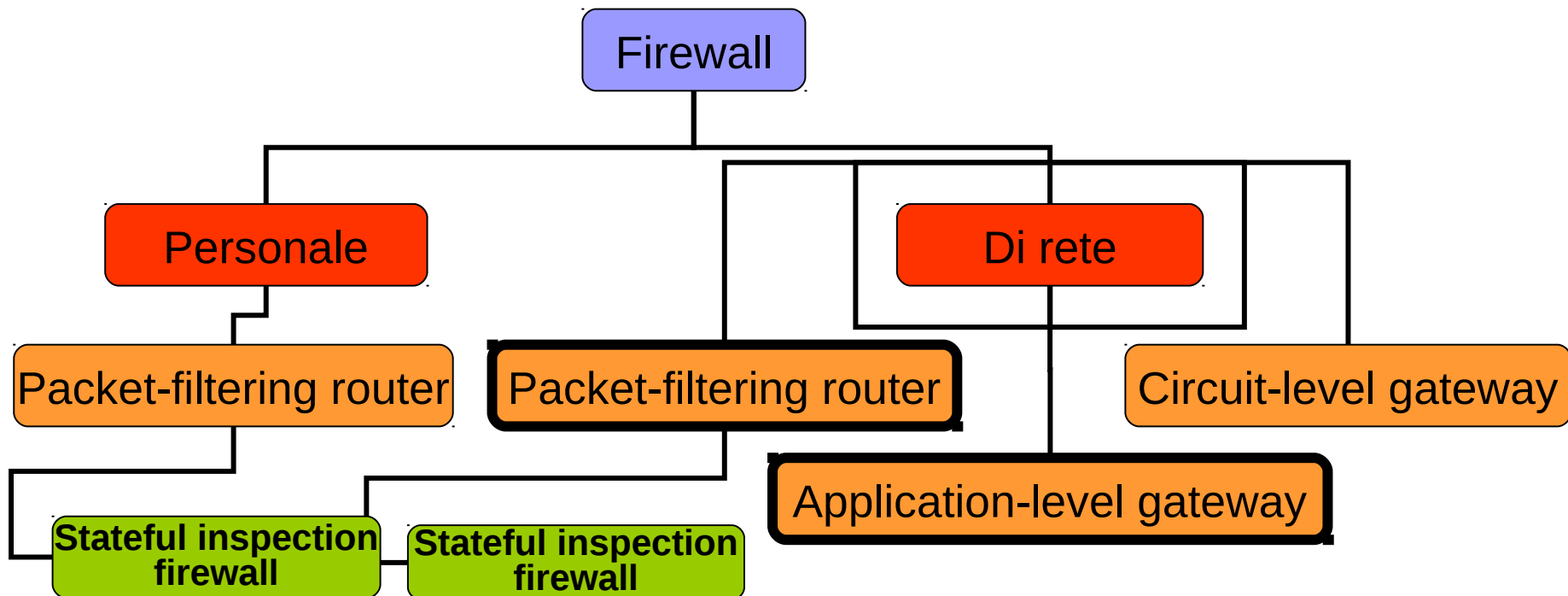
Stateful inspection firewall

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers



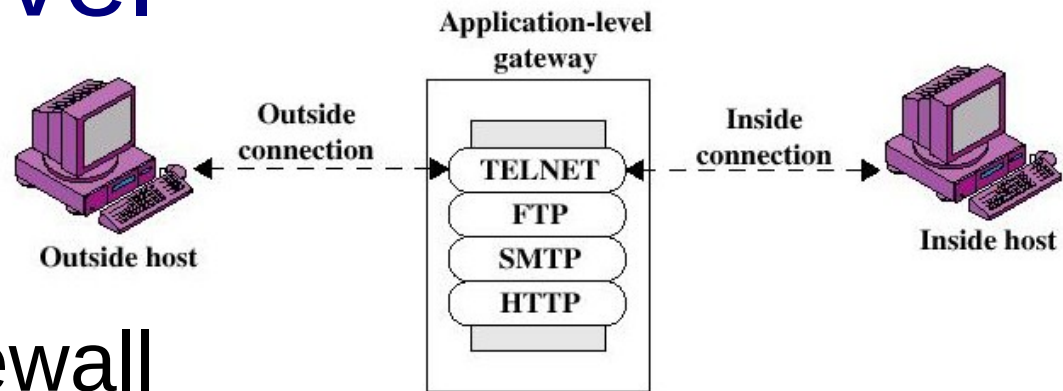
Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Firewall – tassonomia



Application-level gateway

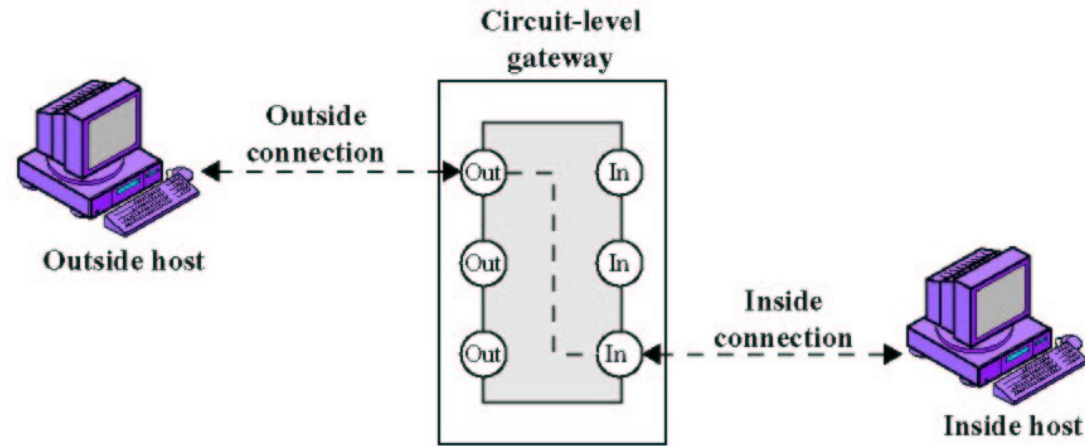
- Anche detto proxy server firewall
- Livello applicazione
- Il software del firewall esegue instradamento
- Gestisce ogni forma di autenticazione utente
 - Conoscenza
 - Possesso
 - Caratteristiche biometriche



Application-level gateway

- Può gestire specifiche funzionalità delle applicazioni
 - *Permettere get, negare put*
- Relativamente facile da configurare
 - Sorverglia solo specifiche applicazioni piuttosto che le varie forme di trasporto
- Monitoraggio efficace
 - Anche su comandi specifici di un'applicazione
- Prezzo computazionale

Circuit-level gateway



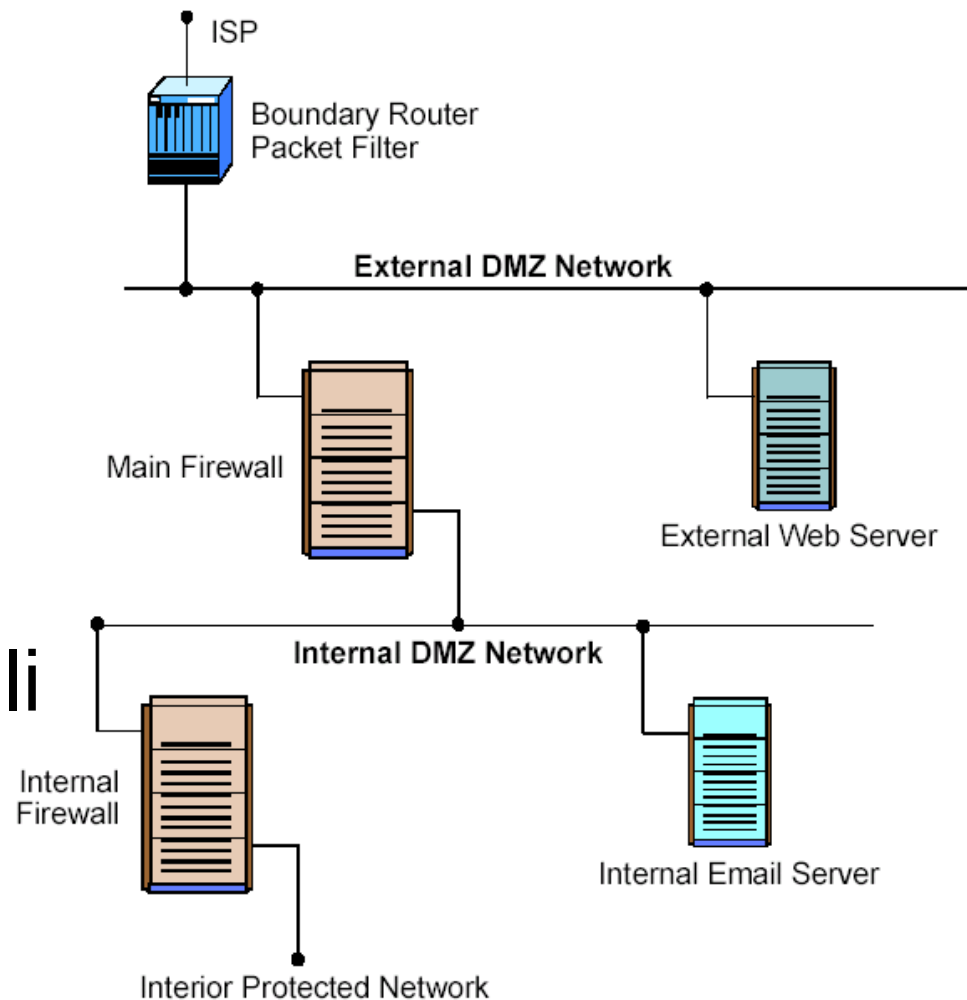
- Livello trasporto
- Non permette connessioni punto-punto
- Apre due connessioni vere e proprie, una col mittente e una col destinatario
- Stabilisce quali pacchetti possano passare da una all'altra

Principi di utilizzo

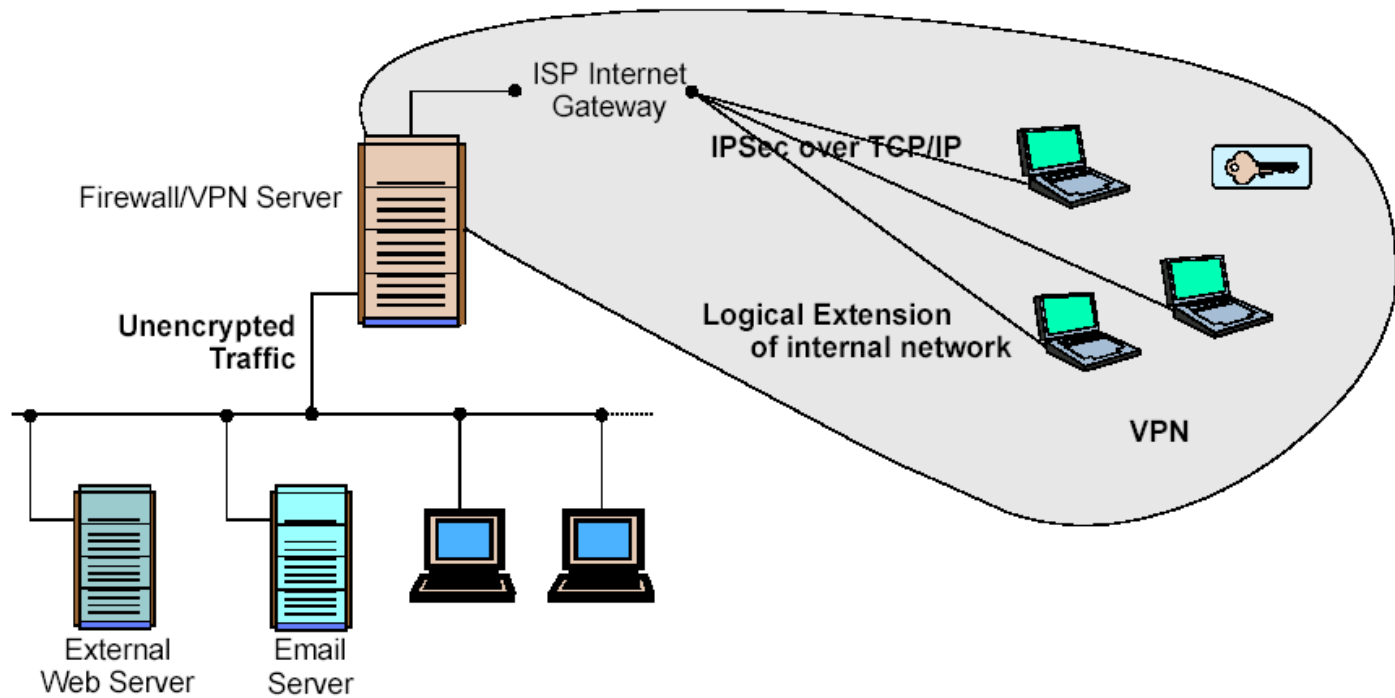
1. KISS (Keep It Simple, Stupid)
 - Non fidarsi di ingarbugliati insiemi di regole
2. Usare ciascun dispositivo per i suoi scopi
 - Firewall, IDS, switch, router
3. Creare difese in profondità
 - Usare più firewall in cascata
4. Non dimenticare mai le minacce interne

Applicazione 1: DMZ

- **DMZ** (Demilitarized Zone Network): rete fra due firewall
- Può ospitare servizi (più o meno) pubblici
- Macchine più sensibili ricevono più livelli di protezione



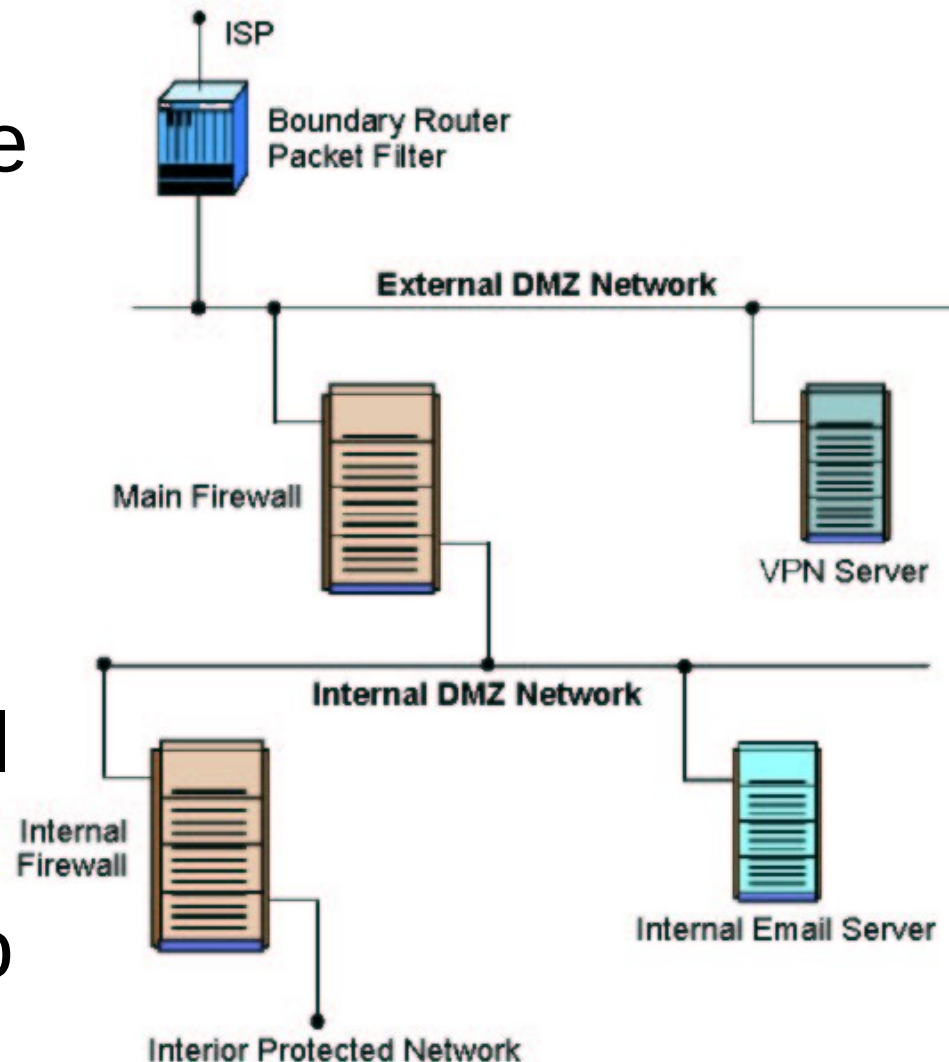
Applicazione 2: VPN



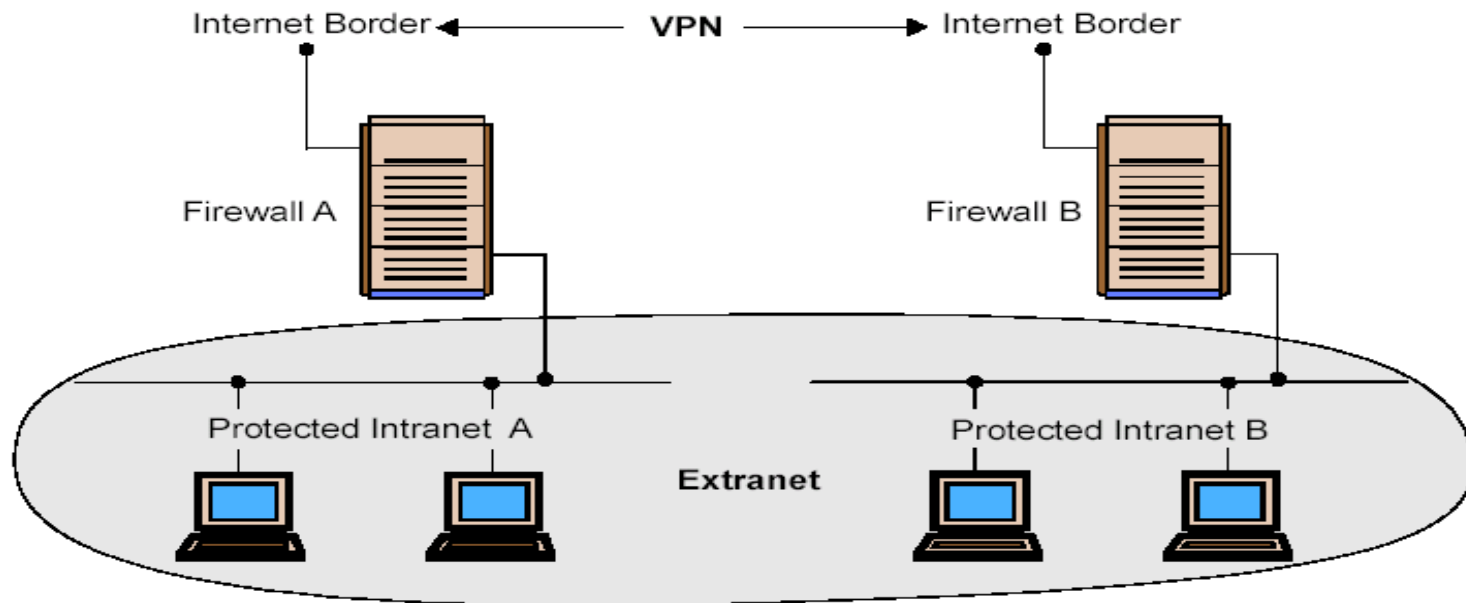
- Un protocollo di sicurezza crea un tunnel sicuro fra un utente mobile e il firewall

Applicazione 3: DMZ + VPN

- Server VPN insieme a firewall principale inficia prestazioni
- Il server VPN può andare in una DMZ
- Il flusso uscente dal server VPN viene ulteriormente filtrato



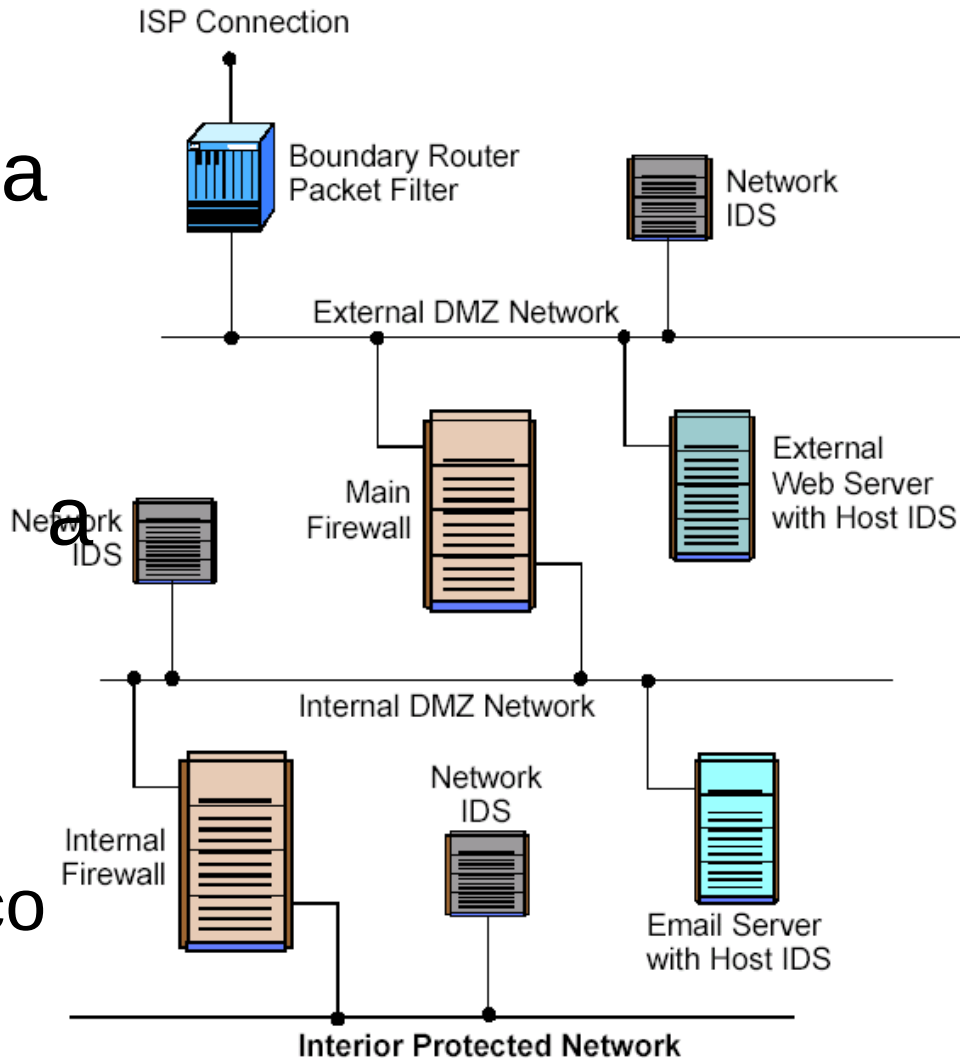
Applicazione 4: Intranet/Extranet



- Extranet: due o più intranet collegate attraverso una VPN su Internet

Applicazione 5: IDS

- Il firewall può essere usato in concomitanza con un IDS (Intrusion Detection System)
- Reazione immediata a tentativi di attacco
 - Se l'IDS rileva un tentativo di DoS, il firewall blocca il traffico dalla sorgente

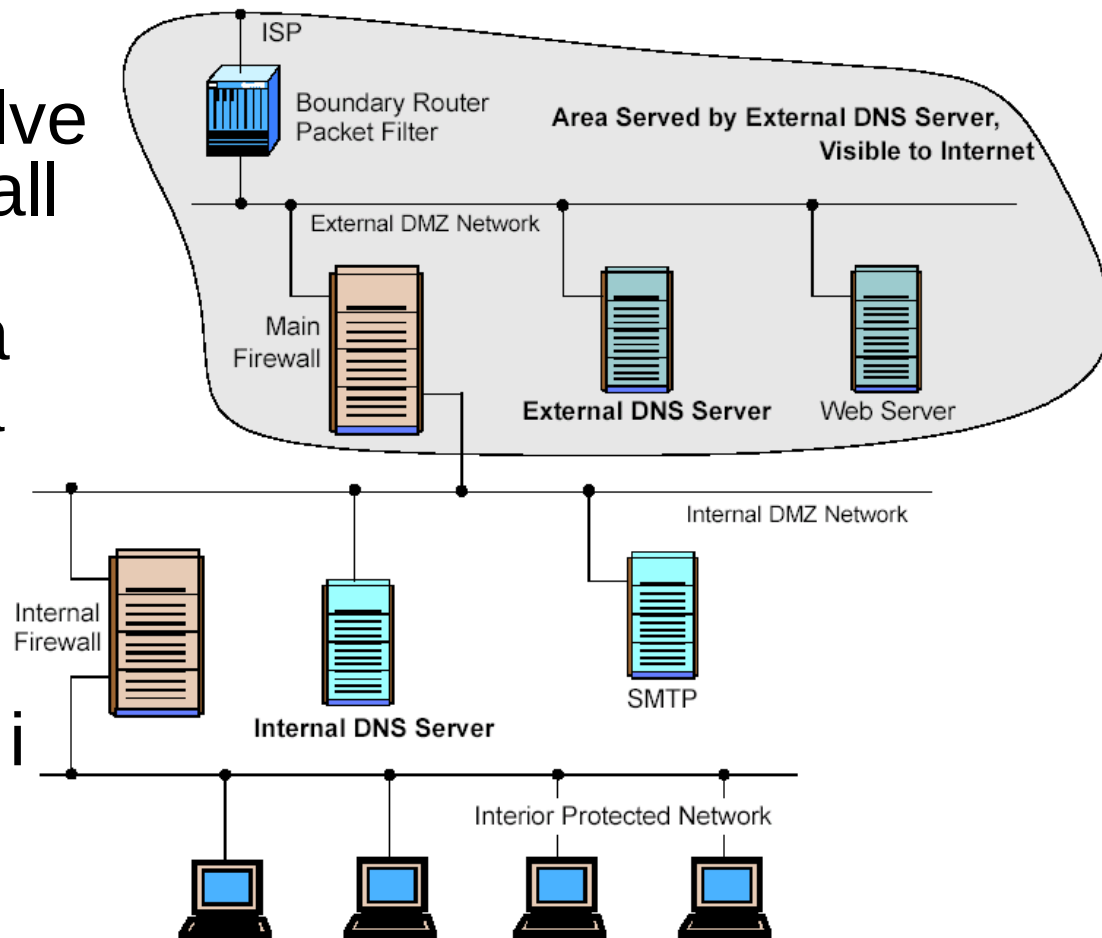


Applicazione 6: DNS

■ 2 DNS separati

□ DNS esterno risolve i nomi per il firewall principale e tutti i nodi esterni (sulla DMZ esterna) ma non per i nodi interni

□ DNS interno risolve i nomi per i nodi interni (sulla DMZ interna) e quelli in Internet

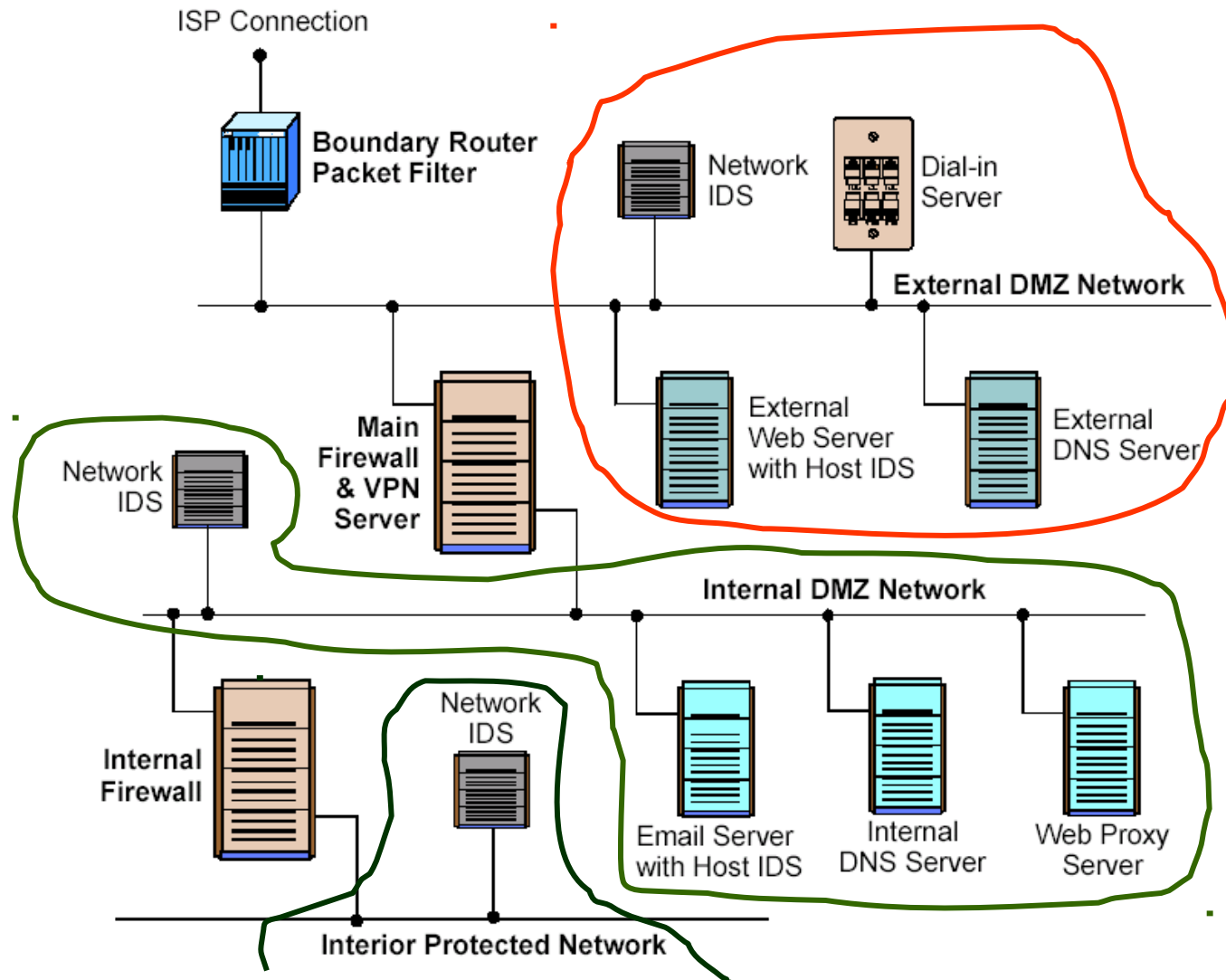


Applicazione 7: Server

■ Regole generali

- ☐ Server esterni protetti da un packet filter router
- ☐ Nessun server accessibile dall'esterno sulla rete interna
- ☐ Usare server interni (al di qua del firewall interno) solo se strettamente indispensabile
- ☐ Server sulla DMZ interna protetti da attacchi provenienti dall'interno

Applicazione 7: Server



Applicazione 7: Server

- Il firewall principale ha solo 3 porte aperte
 - Servizio www (http)
 - Servizio www sicuro (https per VPN)
 - Servizio posta elettronica (SMTP)
- Scandisce le richieste su queste porte alla ricerca di minacce
- Se una richiesta http o SMTP supera i controlli, contatta i relativi proxy sulla DMZ interna

Applicazione 7: Server

- Rimangono 3 strategie d'attacco
 1. Violare porta HTTP
 2. Violare porta SMTP
 3. Sfruttare vulnerabilità delle regole del firewall
- Conclusioni ovvie ☺
 - I daemon relativi a 1 e 2 vanno programmati accuratamente
 - *Banale filtro antispam e antiDoS su SMTP*
 - Le regole del firewall vanno definite accuratamente



Usare un firewall

- Tutti i servizi inattivi devono essere chiusi
- Server pubblici non possono essere chiusi, vanno amministrati
- Servizi/macchine interne possono essere invisibili dall'esterno
- Altri servizi “non amministrati” eventualmente presenti sulle macchine aziendali non sono comunque visibili dall'esterno



Configurazione 0

- Tutto ciò che proviene dall'esterno o dall'interno per il firewall va bloccato
- Tutto ciò che proviene dal firewall va accettato
- Tutto ciò che proviene dall'esterno verso l'interno va bloccato
- Tutto ciò che proviene dall'interno per l'esterno passa indisturbato

Netfilter

- **IpTables** è il firewall standard per i kernel 2.4 di Linux e successivi
- E' solo una parte di un'infrastruttura inglobata nel kernel chiamata Netfilter
 - Realizzata da Rusty Russel (www.netfilter.org)



Netfilter

- Consente di scrivere appositi moduli per gestire il filtraggio o la manipolazione dei pacchetti e di caricarli solamente quando necessario all'interno del kernel



Pregi di IpTables

1. Gira da un 486 in su
2. File di configurazione testuale
3. Manipolazione dei pacchetti in diversi momenti del processo di trasferimento del pacchetto da una scheda ad un'altra
4. Catene



Pregi di IpTables

- 5. Marcatura dei pacchetti IP
- 6. Valido contro i DoS (Rate limiting)
- 7. Passaggio del pacchetto ad una procedura esterna
- 8. Plug-in

Implementazione della politica

- Netfilter tratta i pacchetti in maniera differente rispetto alle precedenti versioni del kernel
- Quando uno di essi arriva dalla rete viene prima di tutto sottoposto ad un processo di routing e quindi se destinato o proveniente dalla rete locale e diretto verso l'esterno è sottoposto solamente al filtraggio della catena **FORWARD**

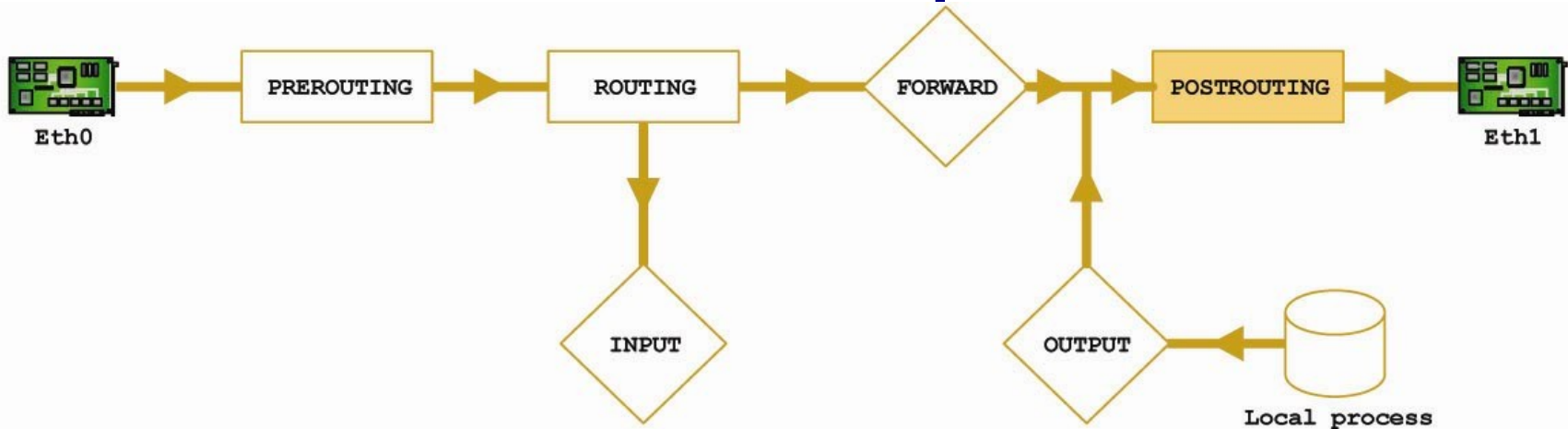
Implementazione della politica

- La catena **INPUT** agisce solamente per tutti i pacchetti destinati esclusivamente al firewall
- La catena **OUTPUT** si occupa di filtrare esclusivamente quelli generati

Implementazione della politica

- Le catene principali accettano di default tutti i pacchetti in transito
 - INPUT: utilizzabile per tutti i pacchetti destinati esclusivamente alla macchina firewall e quindi elaborati dai processi locali
 - FORWARD: Pacchetti destinati ad una delle macchine della rete locale (LAN) o provenienti da essa e dirette all'esterno e non al firewall
 - OUTPUT: Pacchetti generati dalla macchina firewall diretti all'esterno

Funzionamento di IpTables



- Durante i vari processi di trasferimento del pacchetto si possono impostare dei filtri decisionali
- Analogia flusso di liquido in un tubo. Questi hook sono fori di ispezione nel tubo da cui è possibile pescare o anche solo osservare il flusso
- **ACCEPT – DROP – REJECT**

Configurazione di IpTables

1. Prime tre condizioni della configurazione 0

- ☐ **-P INPUT DROP**
- ☐ **-P OUTPUT ACCEPT**
- ☐ **-P FORWARD DROP**

2. La quarta si traduce così

- ☐ **-A FORWARD -s x.x.x.x/n -i eth1 -j ACCEPT**

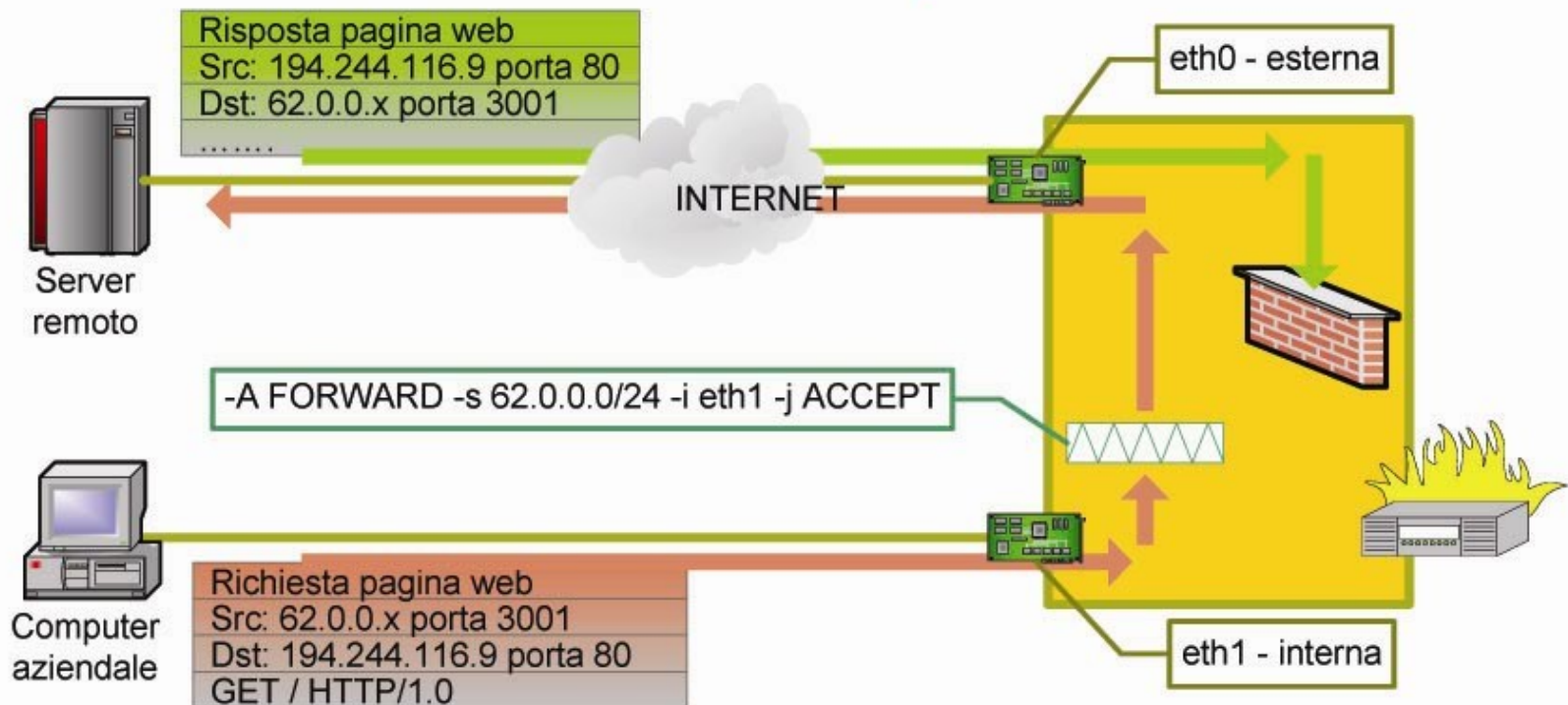
Gli utenti interni non riescono ancora a navigare: i pacchetti di risposta non arrivano



La risposta a richieste interne

- Come mai i pacchetti di risposta non entrano?
 - Sono bloccati dalla regola -P FORWARD DENY
- Tipica connessione di tipo HTTP

La risposta a richieste interne



Soluzione standard (no ipTables)

Aprire all'esterno le porte non privilegiate

- Si fa l'assunzione che queste porte
 - Siano aperte per brevi intervalli di tempo
 - Non abbiano privilegi di sistema elevati
 - Vengano aperte da sistemi client e non server
- Si dimentica che
 - Va contro la filosofia di configurazione del firewall
 - Alcuni servizi leciti sono attivi su queste porte (Proxy su 3180/8080)
 - Porte sfruttabili da software nocivo



Soluzione avanzata (IpTables)

Operazioni per gestire intere catene

1. Crea una nuova catena (-N)
2. Cancella una catena vuota (-X)
3. Cambia la politica di una catena (-P)
4. Elenca le regole in una catena (-L)
5. Svuota una catena delle sue regole (-F)
6. Azzera i contatori dei pacchetti e dei byte di tutte le regole di una catena (-Z)



Soluzione Avanzata (IpTables)

Operazioni per gestire le regole di una catena

1. Appendi una nuova regola alla catena (-A)
2. Inserisci una nuova regola in una determinata posizione della catena (-I)
3. Sostituisci una regola presente in una certa posizione della catena (-R)
4. Cancella una regola dalla catena (-D)

Glossario

- *tabella (-t)*

L'infrastruttura netfilter ha introdotto tre tipi di tabelle che facilitano le impostazioni delle regole, la prima (**filter**) contiene le tabelle preesistenti ed è utilizzata per impostare le regole di filtraggio, la seconda (**nat**) è utilizzata per le regole che riguardano il masquerading mentre l'ultima (**mangle**) viene utilizzata per la manipolazione dei pacchetti come il marcamento e i bit TOS

- *catena (-A)*

Una catena consiste in una policy che stabilisce quali pacchetti devono essere accettati e quali no, tale opzione permette di aggiungere una determinata regola alle catene elencate, sono disponibili anche altre opzioni che permettono di creare (-N) o cancellare (-D) le catene

Glossario

- *protocollo (-p)*

IPTables permette di applicare le proprie regole in base ad un determinato tipo di protocollo, sono disponibili i comuni tcp, udp, icmp

- *interfaccia (-i -o)*

Il filtraggio dei pacchetti può avvenire in base al nome dell'interfaccia di ingresso (-i) oppure di uscita (-o), molto utile quando si dispone di più interfacce di rete sulla stessa macchina

Glossario

- *porta sorgente e/o destinazione (--sport --dport)*

Queste due opzioni permettono di stabilire la porta (tramite il numero oppure il nome) di destinazione (--dport) o di sorgente (--sport) da utilizzare all'interno della nostra policy.

- *regole (-j oppure -jump)*

Questa opzione ci permette di definire, attraverso una delle parole chiave, il destino (chiamato obiettivo) del pacchetto che soddisfa la regola indicata

Le opzioni

1. **ACCEPT**: accetta i pacchetti che soddisfano la regola indicata
2. **DROP**: scarta il pacchetto (sostituito da DENY su ipchains)
3. **REJECT**: scarta il pacchetto e avvisa tramite il messaggio (port unreachable)
4. **QUEUE**: accoda i pacchetti per una successiva elaborazione
5. **MIRROR**: rispedisce il pacchetto alla macchina che lo ha generato
6. **LOG**: registrazione dei pacchetti, obiettivo dotato di ulteriori opzioni
7. **MASQUERADE**: utilizzato per impostare il NAT

Modulo ip_conntrack (Stateful Inspection)

- Apertura delle porte non privilegiate fa da palliativo all'impossibilità pratica di capire, dal solo pacchetto in ingresso, se sia correlato ad una richiesta interna
- Soluzione di IpTables: tracciare le richieste
- Analisi che confronta il pacchetto in ingresso con tutti quelli transitati in uscita
- Questa analisi ritorna un eventuale stato del pacchetto relativamente a pacchetti usciti in precedenza

Stati restituiti da ip_conntrack

1. NEW

- ☐ Pacchetto non correlato. Nuova connessione dall'esterno

2. ESTABLISHED

- ☐ Pacchetto appartenente ad una connessione esistente (caso richiesta www)

3. RELATED

- ☐ Pacchetto correlato ma non appartenente ad una connessione (caso richiesta FTP-FTP data)

4. INVALID

- ☐ Errore nel processo di ricerca

5. La regola che completa la configurazione 0 e' quindi

- ☐ **-A FORWARD -d x.x.x.x/n -m state **
--state ESTABLISHED,RELATED -j ACCEPT

Aprire un servizio pubblico

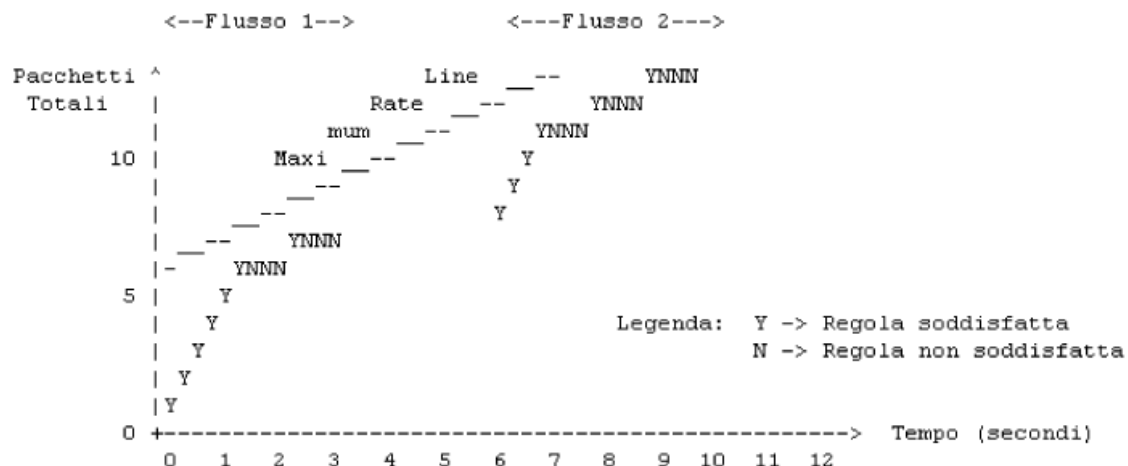
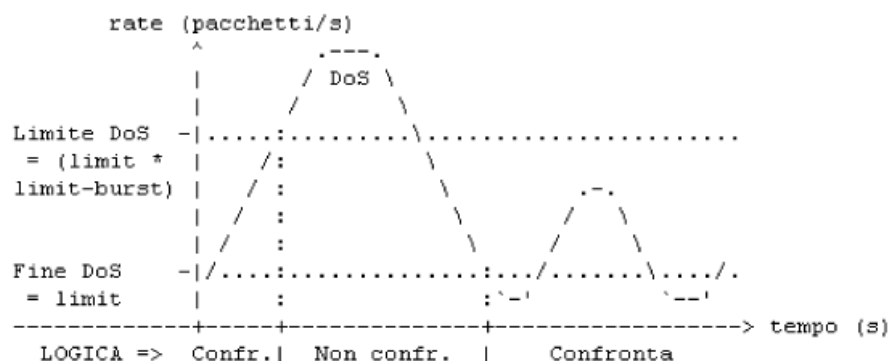
- Server web interno

- **-A FORWARD -p tcp -d y.y.y.y \
--dport web -j ACCEPT**

Limit Bursting e porte isteriche

- Limitare i ping in ingresso

**-t FILTER **
**-A FORWARD **
**-p icmp **
**-icmp-type **
**echo-request **
**-m limit **
**-limit 1/s **
-j ACCEPT



SNAT e masquerade

- **-t nat -A POSTROUTING -o ppp0 -j \ MASQUERADE**
- **-t nat -A POSTROUTING -o eth0 -j SNAT \ -to xxx.xxx.xxx.xxx**

Destination Nat

- **-t nat -A PREROUTING -p tcp -d y.y.y.y \ -
dport web -j DNAT -to z.z.z.z**

Routing avanzato

- **-A PREROUTING -i eth0 -t mangle -p tcp \ -dport web -j MARK --set-mark 1**
- **ip rule add from x.x.x.x table linkFast**
- **Ip route add default via y.y.y.y dev eth1 \ table linkFast**
- **Ip rule add fwmark 1 lookup linkFast**



Riferimenti

1. Linux Networking HowTo
2. Linux 2.4 Packet Filtering HowTo
3. Linux 2.4 Nat HowTo
4. IpRoute2 Utility Suite HowTo
5. Linux 2.4 Advanced Routing HowTo
6. MonMotha's Firewall configuration files
7. Linux firewall: dai una marcia in più alla tua rete
 - <http://www.ebruni.it/>



Parte 7

Software Nocivo (Malware)

Bug versus SW nocivo

- Bug: proprietà inattesa del SW
 - Può essere sfruttato da altro software non necessariamente nocivo
 - Non necessariamente?
 - *E se Win2K caricasse un'applicazione utente sovrascrivendo il security module in memoria?*

I bug sono tipicamente preterintenzionali

Bug versus SW nocivo

- **SW nocivo**: SW scritto con l'**esplicito** scopo di violare la sicurezza di un sistema
 - Non necessariamente sfrutta dei bug
 - Non necessariamente?
 - *Ero sicuro non ci fossero bug ma ahimè ho beccato un virus lo stesso*

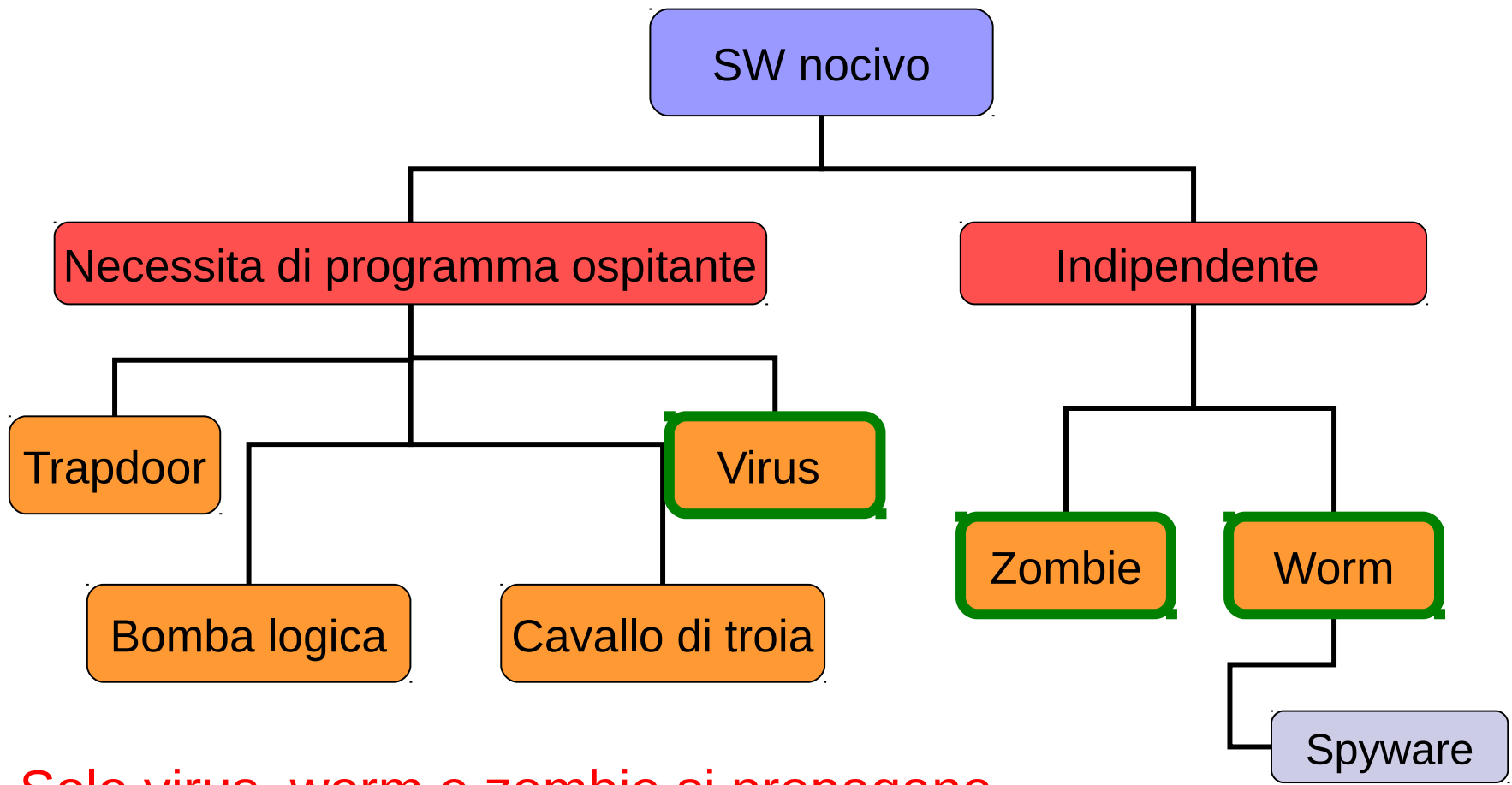
Un bug preterintenzionale non rende nocivo il SW ospitante

SW nocivo – caratteristiche

- **Carico**: specifica violazione di sicurezza
 - ☐ Sempre presente
- **Propagazione**: meccanismo di trasmissione fra le macchine
 - ☐ Non sempre presente

E' più pericoloso il massimo carico o la massima propagazione?

SW nocivo – tassonomia



Solo virus, worm e zombie si propagano

Trapdoor (back door)

Def. Punto segreto di accesso ad un programma senza le procedure di sicurezza altrimenti previste

- Concepito per semplificare beta-testing
 - *Come nel film War Games*
- La trapdoor deve essere nota
 - *Se digiti hrtw eviti la password di 30 caratteri*
 - *Nel biennio 96-98 si poteva telefonare gratis dai telefoni pubblici inglesi*

Bomba logica

Def. Frammento di codice di un programma non nocivo pronto a “esplodere” quando si verificano certe condizioni

- Il più antico tipo di SW nocivo
 - *Ho solo inserito un nuovo utente nel sistema e si è riformattato il disco! L'utente era il 100°*
 - *Omega Engineering perse 10 milioni di dollari a causa di una bomba logica preparata da Tim Lloyd*

Cavallo di troia

Def. Programma utile o apparentemente utile che in fase di esecuzione compia violazioni di sicurezza

- Numerosissimi usi
 - *Rinominare defrag.exe come format.exe*
 - *L'utente che esegue clear si ritrova tutti i propri file con $r-rwxrwx$*
- Come inserirlo nel sistema?
Spesso usate trapdoor

Zombie

Def. Programma nocivo che sfrutta una macchina remota già violata per lanciare nuovi attacchi che difficilmente possono essere ricondotti all'autore dello zombie

- Estremamente diffusi oggi su Internet
- Tipicamente utilizzati per attacchi di negazione del servizio

Worm

Def. Programma nocivo che infetta macchine remote, ciascuna delle quali a loro volta infetta altre macchine remote

- E' più dannoso di uno zombie perché si trasmette da macchina in macchina
- Non richiede intervento umano
 - *Il worm di Morris infettò 6000 macchine in ore*
- Un virus che si propaga via email ha le caratteristiche di un worm

Esempio di worm: Morris

- Worm Morris o worm Internet
- Attacca rete di macchine Unix
- Robert Morris multato \$10.000, 3 anni in carcere (sospeso), 400 ore di servizio civile
- Secondo la storia ufficiale, il worm solamente
 1. Determina dove potersi diffondere
 2. Si diffonde rimanendo non scoperto
 3. Si autoelimina (la sua esecuzione termina)



Worm Morris – carico

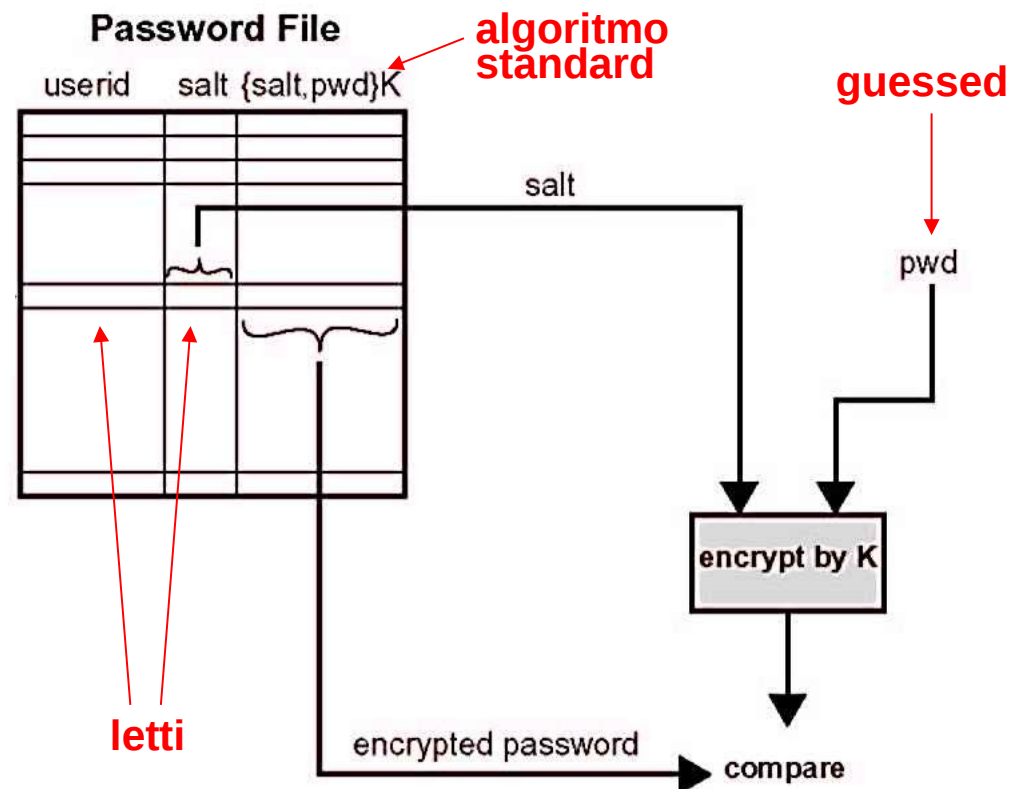
- Secondo la storia ufficiale, Morris ha bug
- A causa dei bug, alcune versioni non terminano, anzi si replicano anche sulla stessa macchina
- Le macchine soffrono serio calo prestazionale
- Migliaia di macchine vengono sconnesse dalla rete o per proteggersi o per proteggere

Worm Morris – propagazione

1. Sfrutta tre **noti** bug di Unix per violare la macchina remota, alternativamente
 - a. Attacco known-ciphertext su **file di password**
 - b. BOF mediante programma **finger**
 - c. Trapdoor nel programma **sendmail**
2. Vi scarica 99 linee di codice C
 - Si auto compilano ed eseguono per poi scaricare l'intero worm

a. Attacco known-ciphertext

- Il file di password è sì criptato ma le sue tre colonne sono leggibili da tutti
- Errore di visibilità nell'implementaz. della politica di sicurezza



a. Attacco known-ciphertext

- Morris prova a codificare possibili candidati
 - Prima, il nome utente e sue permutazioni
 - Se non ha successo, una lista di 432 propri candidati statistici
 - Se non ha successo, tutte le parole nella directory locale /local
- Se ha successo può fare login su una macchina remota per scaricarvi le 99 linee

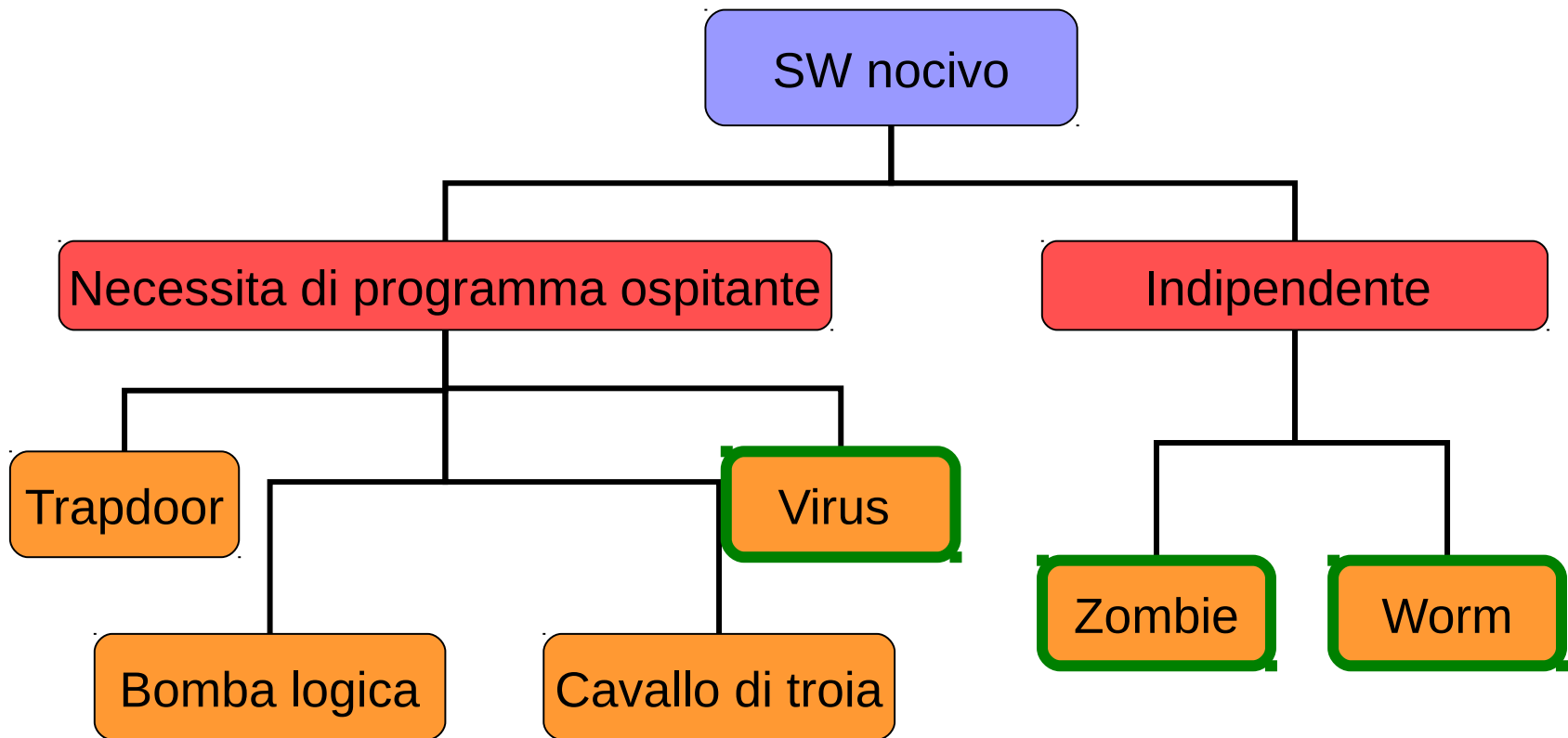
b. BOF mediante finger

- Il worm lancia finger e **fingerd** risponde dalla macchina remota
- BOF sul buffer di input di fingerd, che traborda sull'indirizzo di ritorno
- Fingerd prima di terminare esegue la shellcode caricata nel buffer, la quale scarica le 99 linee

c. Trapdoor in sendmail

- Sendmail è eseguito in background e attende indirizzi email
- Trapdoor: lo si può far passare a “debugging mode” in cui esegue comandi
- Morris sfrutta tale trapdoor sul sendmail remoto per scaricarvi le 99 linee

SW nocivo – tassonomia



Solo virus, worm e zombie si propagano

Virus

Def. Programma nocivo che viola altri programmi non nocivi, sfruttandoli per propagarsi

- Violazione tipica mediante aggiunta di pezzi di codice indesiderati ai programmi
 - Virus per applicativi o per dati
 - Virus per il settore di boot
- Possono essere difficili da individuare
 - Virus polimorfi (cambiano forma)
 - Virus criptati (nascondono le proprie tracce)



Macrovirus

- Virus scritti come macro di un'altra applicazione
 - MS Office: usare le macro?
 - Vari livelli di eseguibilità delle macro
- Possono essere lanciati all'apertura del documento
- Inizialmente infettano senza darne sintomi

SW nocivo – distinzione delicata

	Tipo	Caratteristiche essenziali
Non replic.	Trapdoor	Concede accesso non-autorizzato
	Bomba logica	Si attiva solo su specifiche condizioni
	Cavallo di troia	Ha funzionalità illecite inattese
Replicano	Virus	Attacca un programma e si propaga attraverso qualunque mezzo, anche fisico
	Zombie	Usa macchina già violata per attaccare
	Worm	Viola macchine “ricorsivamente” attraverso la rete

Struttura di un semplice virus

Firma del virus: codice usato per discernere file da infettare da quelli già infetti

Infetta un file ancora “pulito”

Esegue ulteriore danno – per es. una bomba logica

Restituisce il controllo al programma ospitante

```
program V :=  
  {goto main;  
   1234567;  
  
  subroutine infect-executable :=  
    {loop:  
      file := get-random-executable-file;  
      if (first-line-of-file = 1234567)  
        then goto loop  
        else prepend V to file; }  
  
  subroutine do-damage :=  
    {whatever damage is to be done}  
  
  subroutine trigger-pulled :=  
    {return true if some condition holds}  
  
  main: main-program :=  
    {infect-executable;  
     if trigger-pulled then do-damage;  
     goto next;}  
  
  next:  
  
}
```

Dove inserire il virus in un file

- All'inizio
- Alla fine
- All'inizio e alla fine
- Inframezzarlo

Il file infetto si può rilevare perché di lunghezza diversa.
Alcuni virus comprimono il file infetto alla lunghezza originale.

Struttura di un virus che comprime

	program CV :=
1) Comprime il nuovo file da infettare	{goto main; 01234567;
2) Appende il virus all'inizio	subroutine infect-executable := {loop: file := get-random-executable-file; if (first-line-of-file = 01234567) then goto loop;
3) Decomprime il resto del file ospite	(1) compress file; (2) prepend CV to file; }
4) E lo esegue	main: main-program := {if ask-permission then infect-executable; (3) uncompress rest-of-file; (4) run uncompressed file;} }

Esempio di virus: Brain

- Attacca sistemi Microsoft
- Rinomina il volume del disco in BRAIN
- Alcune versioni cancellano frammenti di HD
- Si propaga – ovviamente



Virus Brain – carico

- Caricato in memoria alta, esegue chiamata di sistema per resettare limite di memoria alta al di sotto dell'area che lo contiene
- La posizione 19 del vettore degli interrupt contiene l'indirizzo della procedura per gestire le letture da disco
- Il virus setta quell'indirizzo al proprio, indigerà tutte le letture, anche del boot sector
- In questa prima versione, non crea danni

Virus Brain – propagazione

- Gestendo le letture, controlla se 5° e 6° dei byte letti contengono la sua firma – il valore esadecimale 1234
 - Se NO, infetta 6 settori a caso
 - Se SI', stop – si propaga coi dati



Virus Brain – maggior carico

- Marcare i settori infetti come “danneggiati” cosicchè il S.O. non li usi più
- Iterare il meccanismo di propagazione fino a quando ci sono settori non ancora infetti sul disco
- Lo spazio disponibile sul disco diminuirà continuamente



Rimozione di SW nocivo

1. Antivirus

- Metodo più comune ed in continua evoluzione
- Limitato dalla necessità di aggiornamenti

2. Sistemi immuni

- Prototipo IBM, tardi anni 90', sfrutta antivirus
- Un'intera struttura distribuita di prevenzione

3. Software sentinella

- Integrato col sistema operativo
- Blocca l'esecuzione di codice nocivo

1. Antivirus

Esegue tre compiti

1. **Individuazione**: trova un file con un virus
2. **Identificazione**: stabilisce quale virus sia
3. **Eliminazione**: lo rimuove dal sistema
 - Può essere problematica in caso di virus polimorfi

A seconda di come vengano eseguiti i compiti 1 e 2, si distinguono quattro generazioni di antivirus

1. Antivirus

Prima generazione: confronto con un DB

1. Scansione dei file alla ricerca di firme prese da un DB
2. Scansione di file di applicativi alla ricerca di variazioni nella loro lunghezza rispetto alle lunghezze standard prese da un DB

1. Antivirus

Seconda generazione: uso di euristiche

1. Ricerca di frammenti di codice spesso (statisticamente) associati a virus

- *Il ciclo di compressione di un CV*

2. Ricerca di violazioni di integrità

- *Appendere un checksum ad ogni file di applicativi. Se il file cambia, il checksum dovrebbe cambiare. Ma è codificato con una chiave mantenuta altrove.*

1. Antivirus

Terza generazione: ricerca di azioni illecite

- Programma sempre residente in memoria
- Identifica azioni “illecite”
 - *Quando mai un utente aprirebbe 100 shell al minuto per scopi non nocivi?*
 - *Quando mai un applicativo tenterebbe di cancellare 10GB in una singola richiesta?*
- Blocca un'azione illecita non appena sia rilevabile

1. Antivirus

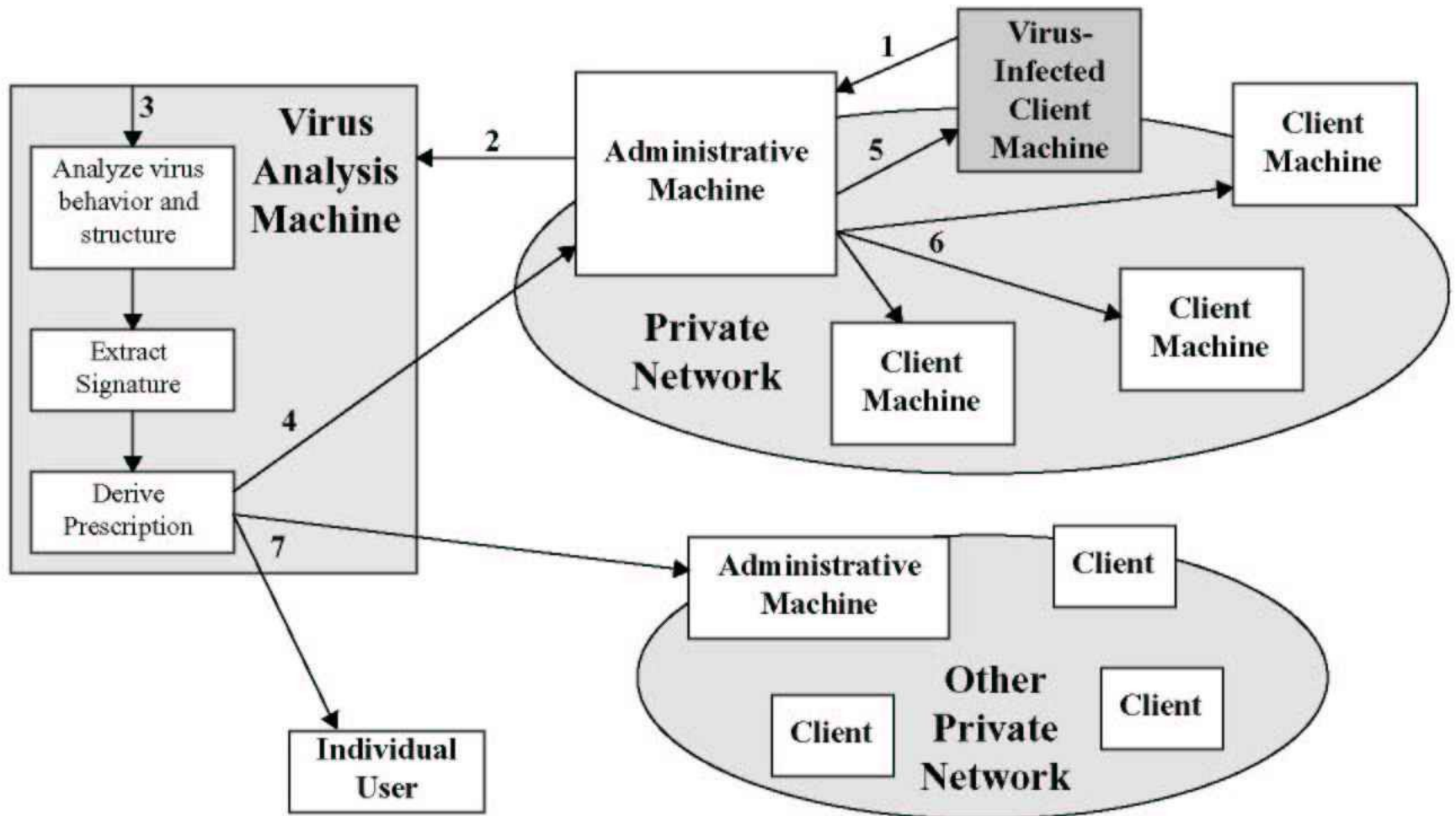
Quarta generazione: insieme di tecniche

- Usa un ventaglio delle prime tre tecniche
- Include capacità di ri-regolare il controllo d'accesso per limitare la propagazione di un virus trovato
 - *Se trova Melissa, impedisce a Internet Explorer di inviare dati e ad Outlook di inviare email*
 - *Trovato un qualunque virus, toglie i permessi di scrittura a qualunque utente*

1. Antivirus – tecniche avanzate

- Usare un **controllore di esecuzione**, che gestisce contemporaneamente:
 1. **Emulatore di CPU**: macchina astratta che verifica le istruzioni di file eseguibili invece di farle eseguire direttamente alla CPU
 2. **Controllore di firme**: modulo che scandisce i potenziali obiettivi alla ricerca di firme di virus; firme note mantenute in un DB da aggiornare

2. Sistemi immuni





2. Sistemi immuni

- Ogni macchina del sistema usa antivirus delle prime tre generazioni
- Una macchina per l'analisi di virus deve essere sempre on-line
- Essa ha tutta la responsabilità di scovare e rimuovere il virus, poi informare i client
- Continuamente aggiornata coi campioni provenienti da tutte le macchine che partecipano all'infrastruttura

3. Software sentinella

- Anche un virus che superi antivirus di qualunque generazione deve alla fine fare “richieste” al sistema operativo
- Un SW sentinella può bloccare (o chiedere conferma all'utente) le richieste potenzialmente pericolose
 - *Tentativi di formattare dischi*
 - *Modifica di impostazioni critiche del sistema*
 - *Inizio di comunicazioni distribuite*

Punti deboli dei browser

1. Gestisce il traffico del client
 - Indica almeno l'indirizzo di ritorno (IP)
2. Gestisce le preferenze di sicurezza del client
 - *Che ci faccio con le applet?* (segue esempio)
3. Mantiene storia e cache delle pagine viste
 - *Ho usato un terminale pubblico in aeroporto...*
4. Possiede le chiavi pubbliche di diverse RCA
 - Gestione delicata

Punti deboli dei browser

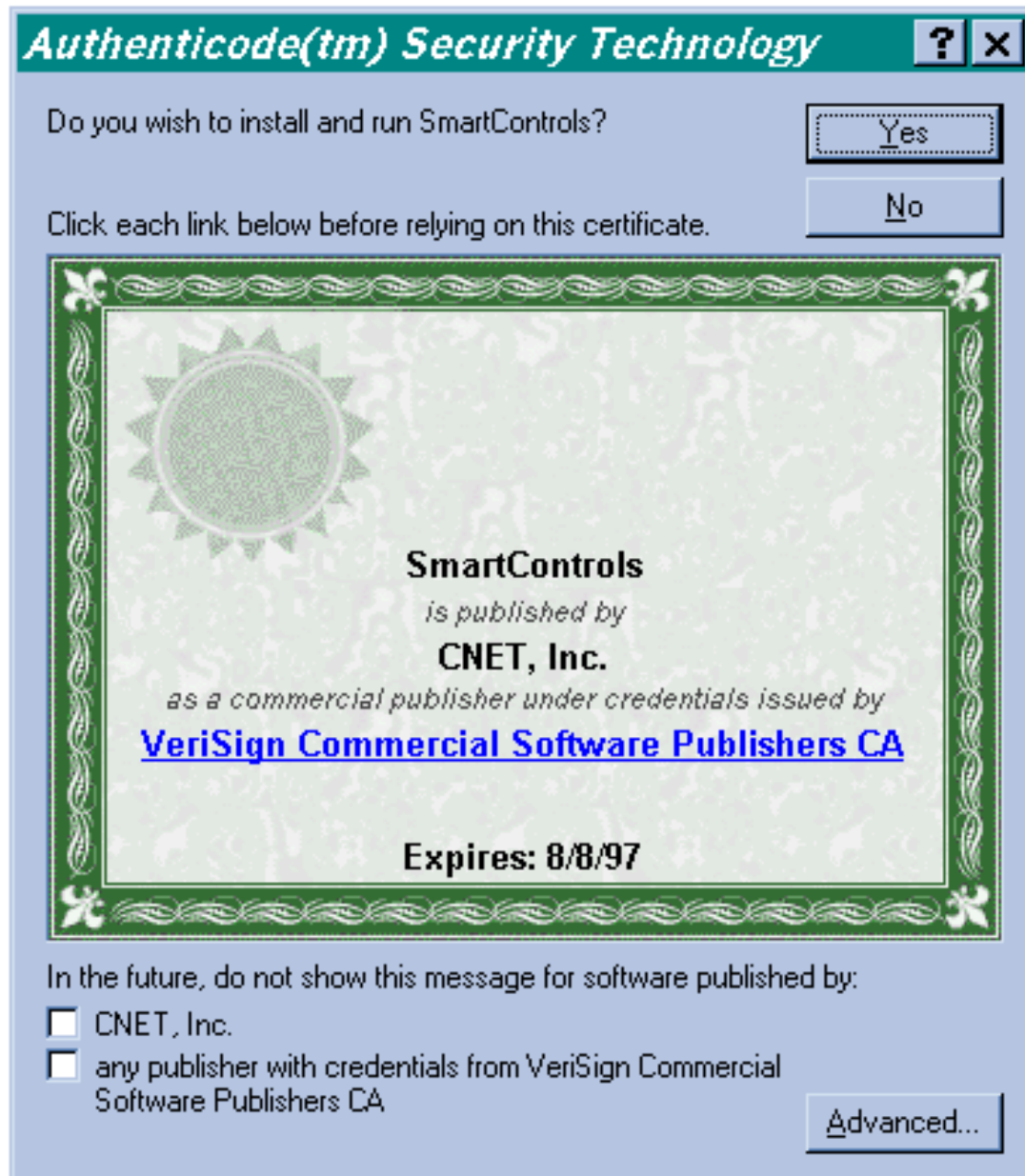
5. E' estremamente modulare
 - ❑ Auto installa continuamente nuovi plug-in
6. Ha spesso accesso a tutte le risorse di sistema
 - ❑ *Non possiamo "concedere" l'intero sistema a Internet!*
7. Memorizza password web dell'utente
 - ❑ Come le protegge?
8. Esegue codice mobile

Rischi dal codice mobile

- Vari tipi di linguaggi di script per i browser
 1. **JavaScript**: di Netscape, consente semplici operazioni come aprire e chiudere finestre
 2. **ActiveX**: firmato digitalmente in PKI Authenticode
 3. **Java**: applet scaricate ed eseguite localmente
- Può essere abusato per numerosi attacchi:
 - Monitoraggio dei siti visitati dagli utenti
 - Lettura dei file locali degli utenti
 - Scansione dell'indirizzario di posta elettronica

Verifica di codice mobile

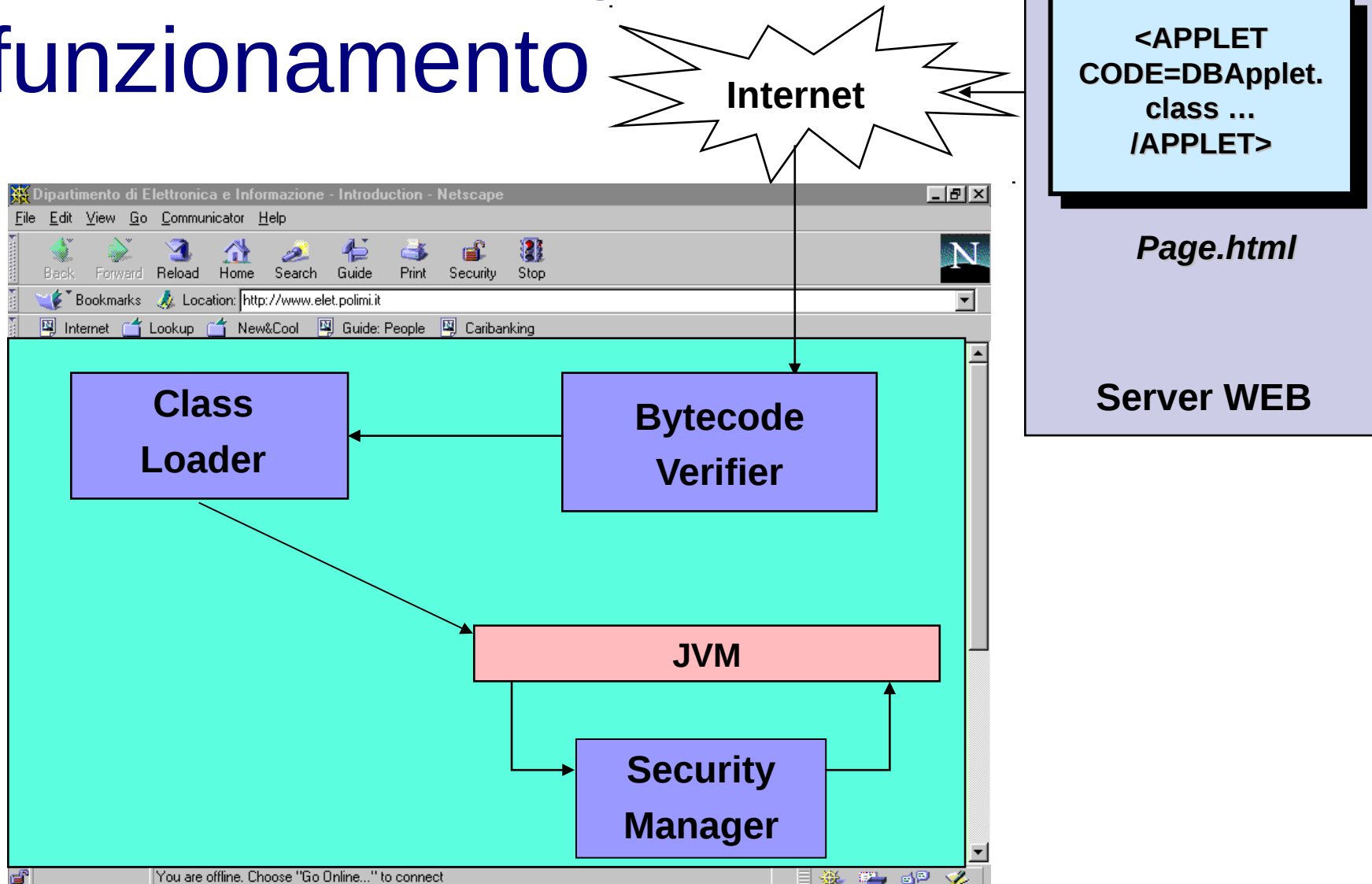
- Sistema
Authenticode
per controlli
ActiveX
- Proprietario M\$



Sandbox di Java

- Richiede che un'applet superi 3 moduli di controllo prima di essere eseguita
- I 3 moduli sono
 1. **Bytecode verifier**: controlla il bytecode Java contenuto nei file *.class
 2. **Class loader**: controlla le classi Java da caricare in memoria
 3. **Security manager**: controlla la correttezza dei metodi secondo un insieme di regole

Sandbox – diagramma di funzionamento





Bytecode verifier

Esegue i seguenti controlli

1. Il file .class sia nel formato corretto
2. Lo stack non vada in overflow
3. Tutti gli operandi abbiano argomenti del giusto tipo (static type safety)
4. Non vi sia conversione di dati da un tipo a un altro
5. Tutti i riferimenti ad altre classi siano legali



Class loader

- Controlla le classi caricate in memoria
- Stabilisce quali possano essere caricate
- Un'applet non può creare il proprio class loader
- Classi provenienti dalla rete vengono gestite con maggiore cautela di quelle built-in




Parte 9

SSL

SSL

- Protocollo di sicurezza fra livello trasporto e livello applicazione
 - Ma sia SSL che TLS possono essere implementati ad ambedue i livelli
- $\text{HTTPS} \approx \text{HTTP} + \text{SSL}$
- Il più usato in rete



Cosa SSL garantisce

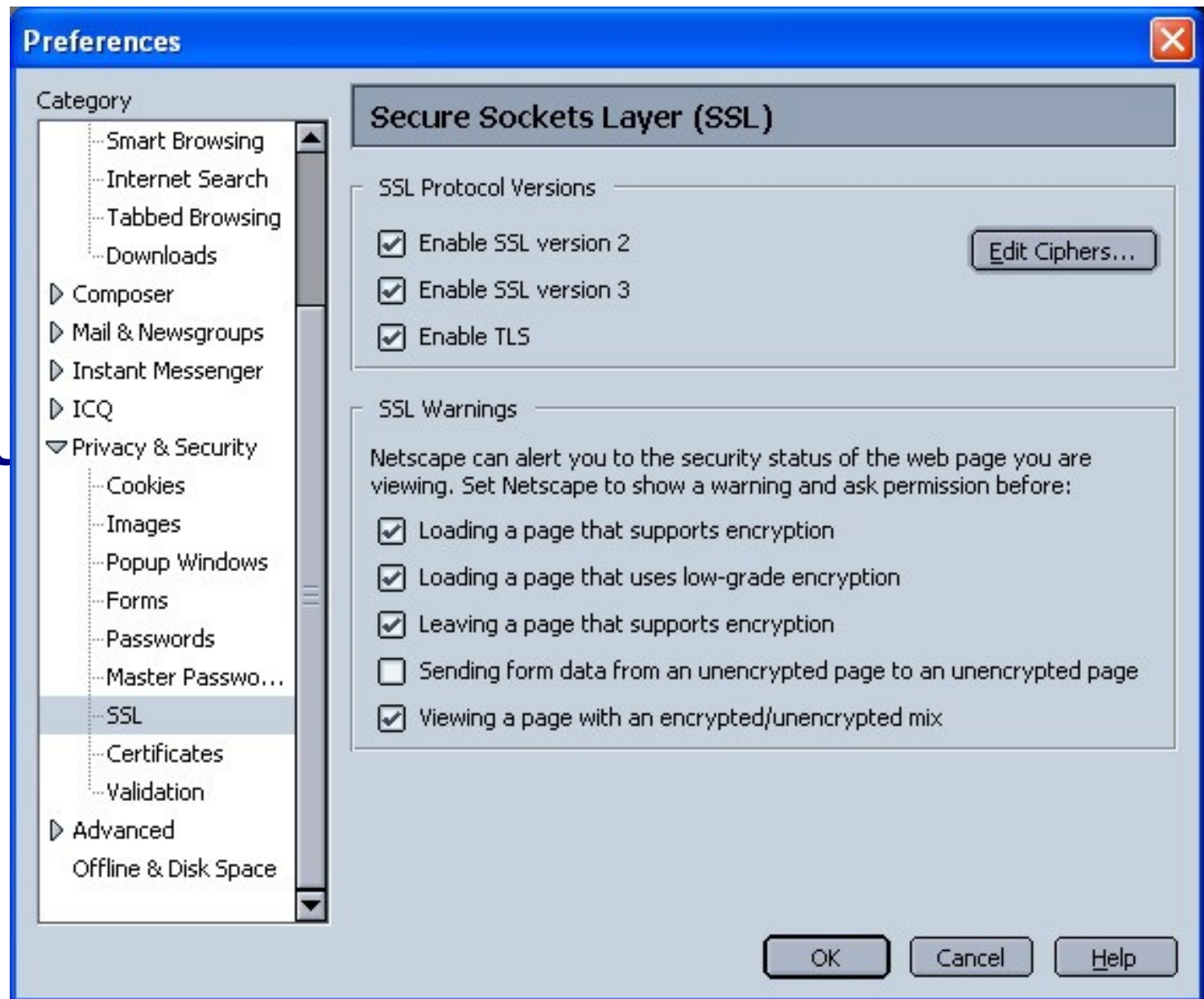
- Segretezza: con l'uso di codifica simmetrica
- Integrità: con l'uso di codifica asimmetrica e hashing
- Autenticazione (utente): con l'uso di firma elettronica e certificazione per il server e, opzionalmente per il client

Come scoprire se uso SSL

- Identificato da HTTPS, FTPS, SMTPS
- Viceversa, se SSL è abilitato, può essere usato indirizzando https://...
- Su una connessione HTTP, l'URL diventa automaticamente https://...
 - Il “lucchetto” si chiude ma Javascript...
- Usa porta 443 invece di 80 come HTTP

Come ottenere SSL

- Pre-installato nei moderni browser, insieme a un insieme di certificati di RCA
- Va abilitato sul server durante il processo di configurazione
 - Semplice su M\$ IIS
 - Compilazione dedicata su Apache



SSL versus TLS

- Netscape lancia SSL
 - SSL1.0 – violato durante la presentazione
 - SSL2.0 – soggetto ad attacco MiM
 - SSL3.0 – versione più recente, metà anni '90
- IETF forma un gruppo di lavoro su TLS
- TLS1.0 va visto come una sorta di SSL3.1
- Adesso TLS è uno standard Internet



OpenSSL

- Implementa SSL/TLS
- Applicazione open source in costante sviluppo
- Permette di gestire certificati digitali
- Permette di ispezionare un'implementazione dei concetti che stiamo per vedere su SSL

OpenSSL: richiesta certificati

```
openssl req -new -newkey rsa:512 -nodes  
-keyout Key.pem -out Req.pem  
-config openssl.cnf
```

- new: indica nuova richiesta
- newkey: specifica formato chiave
- noDes: indica di salvare chiave privata in chiaro
- keyout: indica file con chiave privata
- out: indica file col certificato
- config: indica file con configurazioni di default

OpenSSL: rilascio certificati

```
openssl ca -policy policy_anything  
-out cert.perm -config openssl.cnf  
-infiles req.pem
```

- ca: opzione per la firma
- policy: indica politica da utilizzare per il rilascio
- out: indica file col certificato
- config: indica file con configurazioni di default
- infiles: indica file con la richiesta

OpenSSL: frammento del client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <openssl/ssl.h>
#define LEN 1024
#define PORT 3000

main(int argc, char**argv){
    char  CAfile[]="CACert.pem";
    char  buff_out[]="Prova di invio sicuro";
    char  buff_in[LEN];
    SSL   *ss;
    SSL_CTX *ctx;
    ...    ...
```

Connessione versus sessione

- **Connessione**: forma di trasporto per un determinato tipo di servizio
- **Sessione**: associazione fra un client e un server. Definisce i propri parametri di sicurezza
 - Simile ad SA in IPSec
- Una sessione può realizzarsi mediante più connessioni. Una connessione riguarda una ed una sola sessione



Stato di una sessione SSL

1. **Session identifier**: sequenza di byte arbitrariamente scelti dal server
2. **Peer certificate**: certificato X.509 del nodo
3. **Compression method**: specifica l'algoritmo di compressione applicato prima della codifica
4. **CipherSuite**: specifica l'algoritmo di crittografia e l'hash usato
5. **MasterSecret**: 48 byte condivisi da client/server
6. **Is resumable**: vero se la sessione può essere ripristinata in nuove connessioni



Stato di una connessione SSL

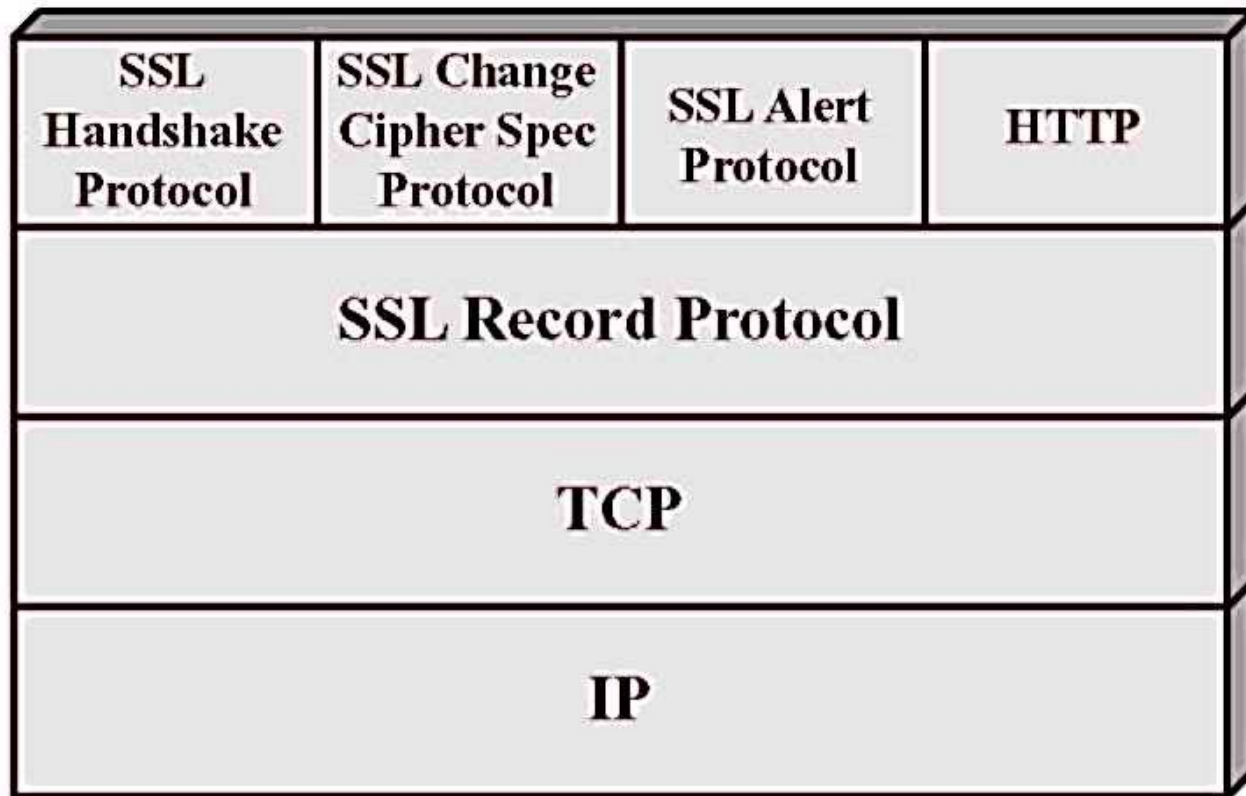
1. **Server and client random**: sequenza di byte scelti da client e server per ciascuna connessione
2. **Server MAC write secret**: chiave per MAC usata dal server
3. **Client MAC write secret**: chiave per MAC usata dal client



Stato di una connessione SSL

4. **Server write key**: chiave di codifica simmetrica usata dal server
5. **Client write key**: chiave di codifica simmetrica usata dal client
6. **Initialization vectors**: usato per inizializzare la codifica a blocchi
7. **Sequence numbers**: numeri sequenziali per i messaggi

I protocolli in SSL





SSL Handshake Protocol (SSL HP)

- Il protocollo più complicato di SSL
- Permette mutua autentica fra client e server
- I quali negoziano chiavi crittografiche, algoritmi di codifica e hashing
 - Stabilito **MasterSecret** da cui creare altri segreti
- Viene eseguito sempre prima che i dati applicazione vengano trasmessi

SSL HP – formato dei messaggi

- **Type**: uno fra 10, come da prossimo lucido
- **Length**: del messaggio in byte
- **Content**: parametri del messaggio, come da prossimo lucido



Handshake Protocol

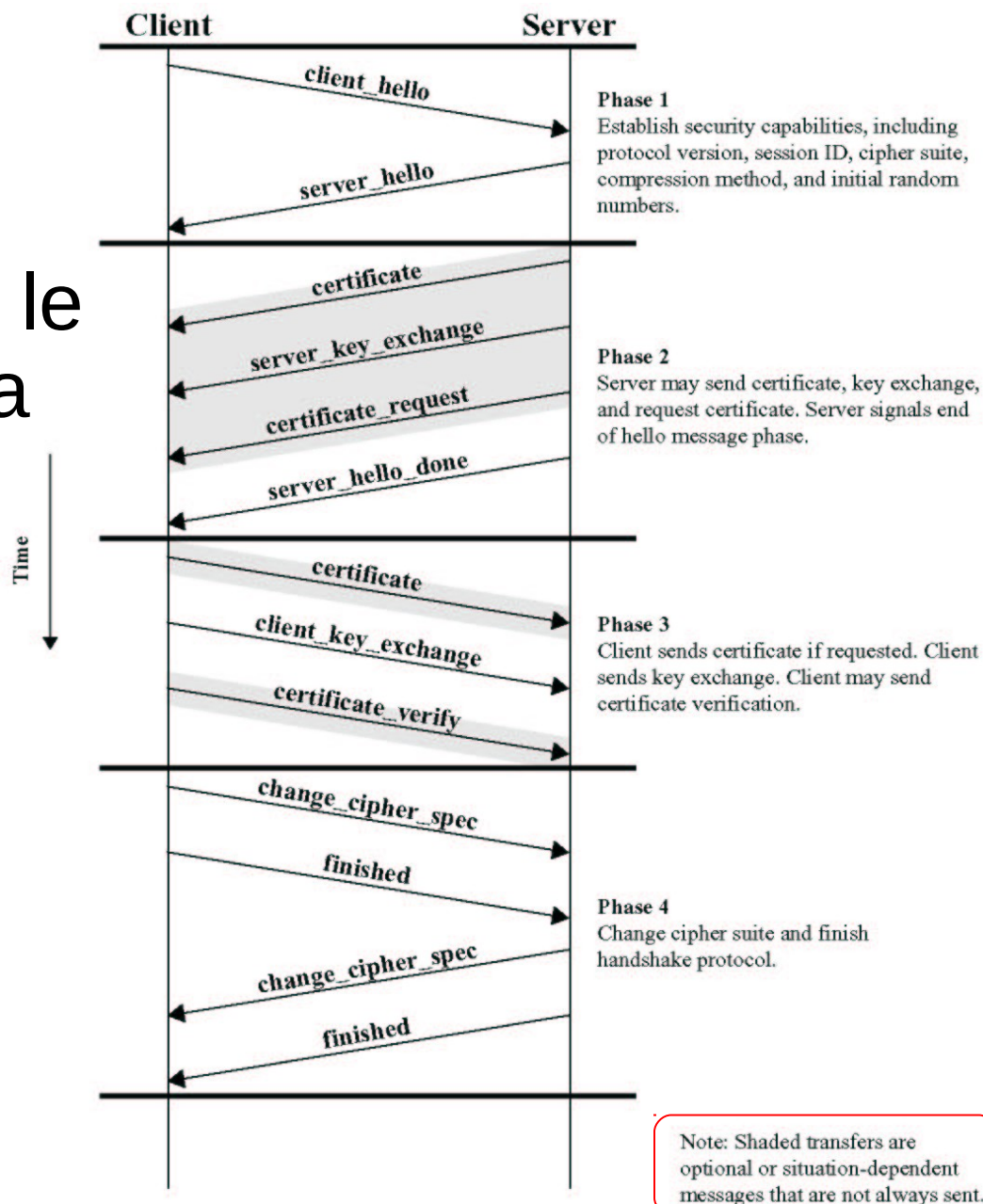
SSL HP – tipi e parametri

SSL Handshake Protocol Message Types

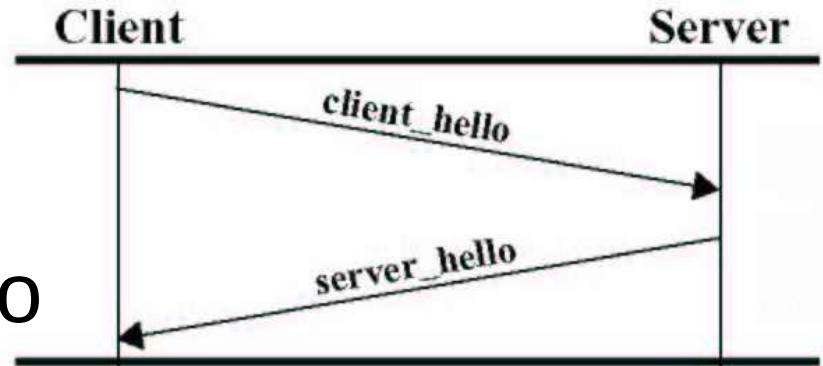
Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

SSL HP

- **Fase 1:** stabilisce le scelte di sicurezza
- **Fase 2:** autentica il server e inizia lo scambio chiavi
- **Fase 3:** autentica il client e termina lo scambio chiavi
- **Fase 4:** conclude



SSL HP – Fase 1



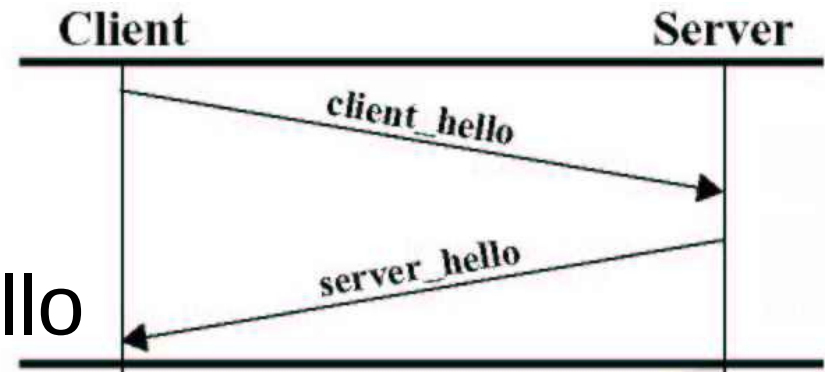
■ Parametri di client_hello

- **Version**: più alta versione supportata dal client
- **Random**: timestamp (32 bit) + nonce (28 byte)
- **Session ID**: uguale a (diverso da) 0 se il client vuole nuova connessione su nuova (medesima) sessione
- **CipherSuite**: lista di preferenze crittografiche
 - Quale protocollo di scambio chiavi, etc.
- **Compression Method**: lista di algoritmi di compressione

SSL HP – Fase 1

■ Parametri di server_hello

- **Version**: minimo fra la proposta del client e la più alta supportata dal server
- **Random**: il server genera il proprio
- **Session ID**: se il client l'ha mandato diverso da zero, il server lo ricopia, altrimenti lo genera
- **CipherSuite**: scelta del server fra le proposte dal client
- **Compression Method**: idem



SSL HP – Fase 2



- **certificate**: il server manda il proprio certificato e la relativa lista di X.509
- **server_key_exchange**: parametri pubblici, pars, per (variante di) Diffie-Hellman, firmati
 - $H(\text{client_hello.random}, \text{server_hello.random}, \text{pars})$
 - Le porzioni nonce prevengono replay attack
- **certificate_request**: richiesta opzionale al client, con lista di autorità accettate
- **server_hello_done**: server OK, client verifica

SSL HP – Fase 3

- certificate: client manda il proprio certificato, oppure no_certificate
- client_key_exchange: client
 - completa (variante di) protocollo D-H, poi calcola chiave condivisa **PreMasterSecret**, oppure
 - genera e invia **PreMasterSecret** di 48 byte codificata con chiave pubblica del server
- Server può calcolare da sé oppure riceve PreMasterSecret



SSL HP – Fase 3

- `certificate_verify`: hash del traffico visto (usa **MasterSecret**)
 - $H(\text{MasterSecret}, \text{pad2}, H(\text{handshake_messages}, \text{MasterSecret}, \text{pad1}))$
- `handshake_messages`: tutti i messaggi da `client_hello` escluso `certificate_verify`
 - Contro forme di replay attack
- Non esiste analogo messaggio dal server

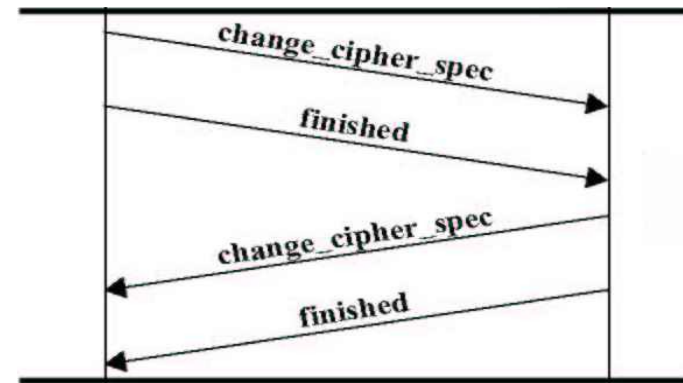


Protocollo di scambio chiavi

- SSL ammette Diffie-Hellmann o sue varianti, oppure protocollo di scambio RSA per produrre PreMasterSecret
- Si espleta con i messaggi `server_key_exchange` e `client_key_exchange`
- Scambio RSA senza `server_key_exchange`
 1. Client → Server : $\{A, \text{PreMasterSecret}\}_{K_b}$
 2. Client e Server calcolano offline $\text{MasterSecret} = f(\text{PreMasterSecret})$

SSL HP – Fase 4

- `change_cipher_spec`:
SSL CCSP per attivare le scelte da Fase 1
- `finished`: concatenazione di due hash
 - `MD5(MasterSecret, pad2, MD5(handshake_messages, sender, MasterSecret, pad1)), SHA(MasterSecret, pad2, SHA(handshake_messages, sender, MasterSecret, pad1)),`
 - `handshake_messages`: totale integrità sessione



MasterSecret (MS)

- MS calcolata da PreMasterSecret (PMS)

- MS =

- MD5(PMS, SHA("A",PMS,ClientHello.random,
ServerHello.random)),
 - MD5(PMS, SHA("BB",PMS,ClientHello.random,
ServerHello.random)),
 - MD5(PMS, SHA("CCC",PMS,ClientHello.random,
ServerHello.random)),

- ...

- Algoritmo iterato fino ad ottenere 48 byte

Generazione delle chiavi

- Dal MasterSecret vengono calcolati i segreti necessari
 - ☐ Client write MAC secret
 - ☐ Server write MAC secret
 - ☐ Client write key
 - ☐ Server write key
 - ☐ Initialization vectors
- Codifica dati (dopo SSL) è simmetrica, quindi veloce

Generazione delle chiavi

- I bit necessari sono prodotti iterando il consueto procedimento a partire da MS
 - `key_block =`
MD5(`MS`, SHA(`"A"`,`MS`,`ClientHello.random`,
ServerHello.random)),
MD5(`MS`, SHA(`"BB"`,`MS`,`ClientHello.random`,
ServerHello.random)),
MD5(`MS`, SHA(`"CCC"`,`MS`,`ClientHello.random`,
ServerHello.random)),
...

Ripristino di sessione

- Sia eseguita una sessione del protocollo su un'unica connessione, client e server memorizzano lo stato della sessione
 - session identifier, peer certificate, compression method, CipherSuite, MS, Is resumable
- Se Is resumable, client e server possono ripristinare la sessione aprendo una nuova connessione sullo stesso session identifier



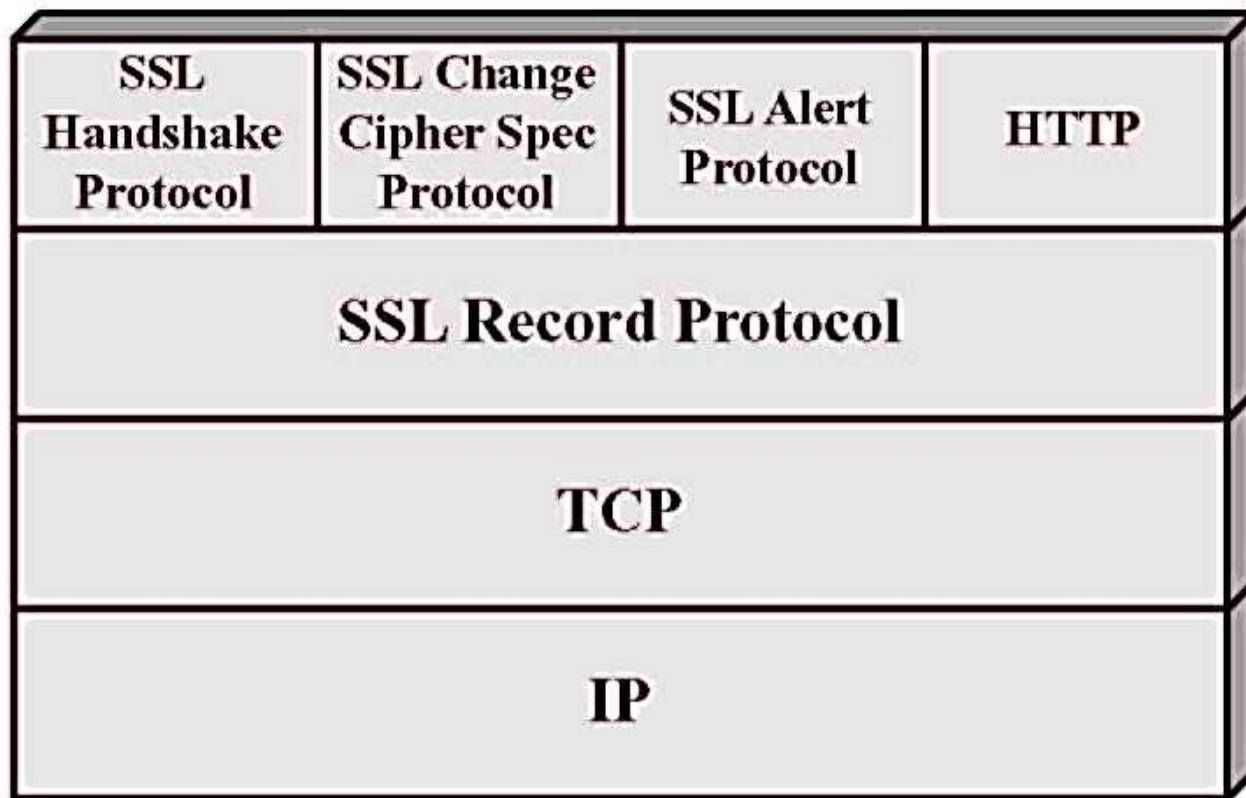
Ripristino di sessione

- La nuova connessione prevede solo Fase 1 e Fase 4
- Fase 1: scambio di nuovi segreti (random) e preferenze crittografiche
- Fase 4: generazione di nuove chiavi usando l'MS già memorizzato
- Pensabile anche il concetto di “connection resumption”

Ripristino di sessione

- Chi la decide?
 - Il client mandando Session Id non nullo in client_hello
- Client e server mantengono il proprio database di Session Id
 - Per quanto tempo?
- Ricevuto Session Id non nullo, il server lo ricerca nel proprio database e decide se farne accettarne il ripristino

I protocolli in SSL



SSL Change Cipher Spec Protocol (SSL CCSP)

- Per cambiare l'algoritmo crittografico in uso
 - L'algoritmo è chiamato "strategia" nell'RFC
- Consiste in due soli messaggi: client e server si scambiano il byte per il valore 1
- Ciascun messaggio segnala che nuovi algoritmi e chiavi vanno posti in uso
 - Scambio tipicamente alla fine dell'Handshake
- Nuovo algoritmo e chiavi negoziati nell'Handshake Protocol

1 byte

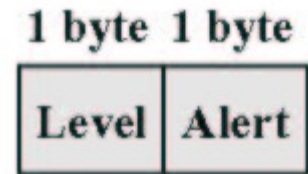
1

SSL Alert Protocol (SSL AP)

- Serve per trasmettere messaggi d'allarme
- Ogni messaggio composto da due byte

- **Level**: livello d'allarme

- **Alert**: quale allarme



- 2 livelli d'allarme

- **Fatal**: la connessione termina, non se ne possono aprire altre sulla sessione

- **Warning**: problema risolvibile

Alert Protocol

SSL AP – fatal alerts

1. **unexpected_message**: messaggio inatteso
2. **bad_record_mac**: MAC incorretto
3. **decompression_failure**: la funzione di decompressione ha ricevuto input improprio
4. **handshake_failure**: il server non è riuscito a negoziare i parametri di sicurezza
5. **illegal_parameter**: un campo di un messaggio di handshake è inconsistente

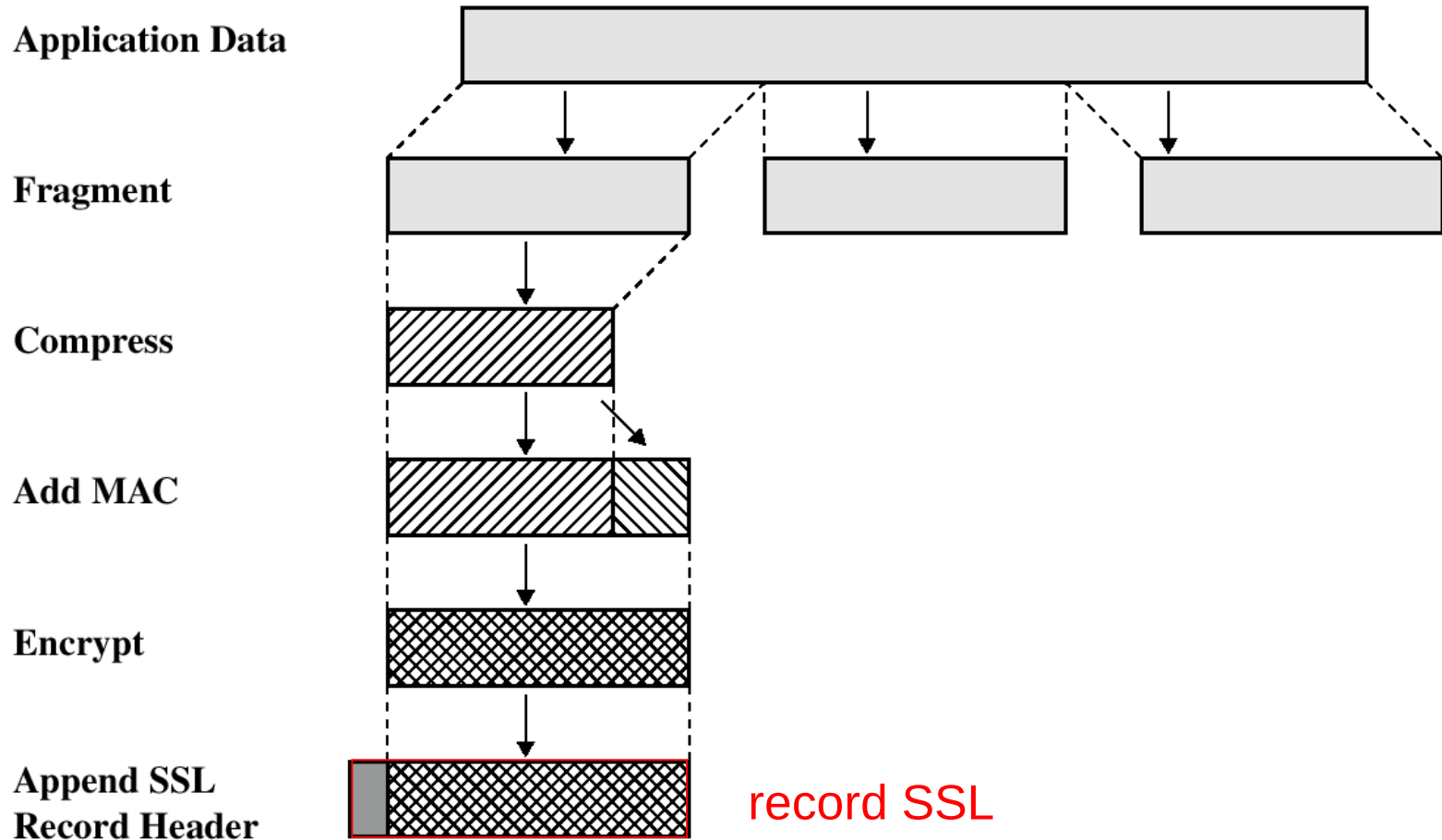
SSL AP – warning alerts

1. **close_notify**: notifica di chiusura connessione
2. **no_certificate**: certificato richiesto indisponibile
3. **bad_certificate**: certificato ricevuto corrotto
4. **unsupported_certificate**: tipo non supportato
5. **certificate_revoked**: revocato dall'autorità
6. **certificate_expired**: certificato scaduto
7. **certificate_unknown**: certificato sconosciuto

SSL Record Protocol (SSL RP)

- Struttura nota: usa chiavi negoziate ai livelli superiori (Handshake Protocol) per ottenere
 - Segretezza mediante codifica simmetrica
 - Autenticazione/integrità mediante MAC

SSL RP – schema di funzionam.



SSL RP – funzionamento

1. **Fragment**: ogni messaggio frammentato in blocchi di 2^{14} byte
2. **Compress**: algoritmo di compressione opzionalmente applicato
 - Nemmeno TLS in realtà lo impone
3. **Add MAC**: MAC ricorsivo, un hash che include un altro hash
 - usa chiave condivisa `MAC_write_secret` ottenuta da livello superiore

SSL RP – MAC

```
H(MAC_write_secret, pad2,  
  H(MAC_write_secret, pad1, seq_num,  
    SSLCompressed.type,  
    SSLCompressed.length,  
    SSLCompressed.fragment))
```

- Se non c'è compressione
 - `SSLCompressed.type = none`
 - `SSLCompressed.length = SSL.length`
 - `SSLCompressed.fragment = SSL.fragment`

SSL RP – MAC

```
H(MAC_write_secret, pad2,  
  H(MAC_write_secret, pad1, seq_num,  
    SSLCompressed.type,  
    SSLCompressed.length,  
    SSLCompressed.fragment))
```

- H: MD5 o SHA-1
- pad1: byte 0x36(00110110) ripetuto 48 volte per MD5, 40 volte per SHA-1
- pad2: byte 0x5C(01011100) ripetuto 48 volte per MD5, 40 volte per SHA-1

SSL RP – funzionamento

4. **Encrypt**: usa chiave condivisa per codifica simmetrica su MAC + testo in chiaro

Block Ciphers

Stream Ciphers

Algoritmo	Lunghezza chiave	Algoritmo	Lunghezza chiave
IDEA	128	RC4-40	40
RC2-40	40	RC4-128	128
DES 40	40		

SSL RP – funzionamento

5. **Append SSL Record Header**: aggiunto in testa un header di 4 campi



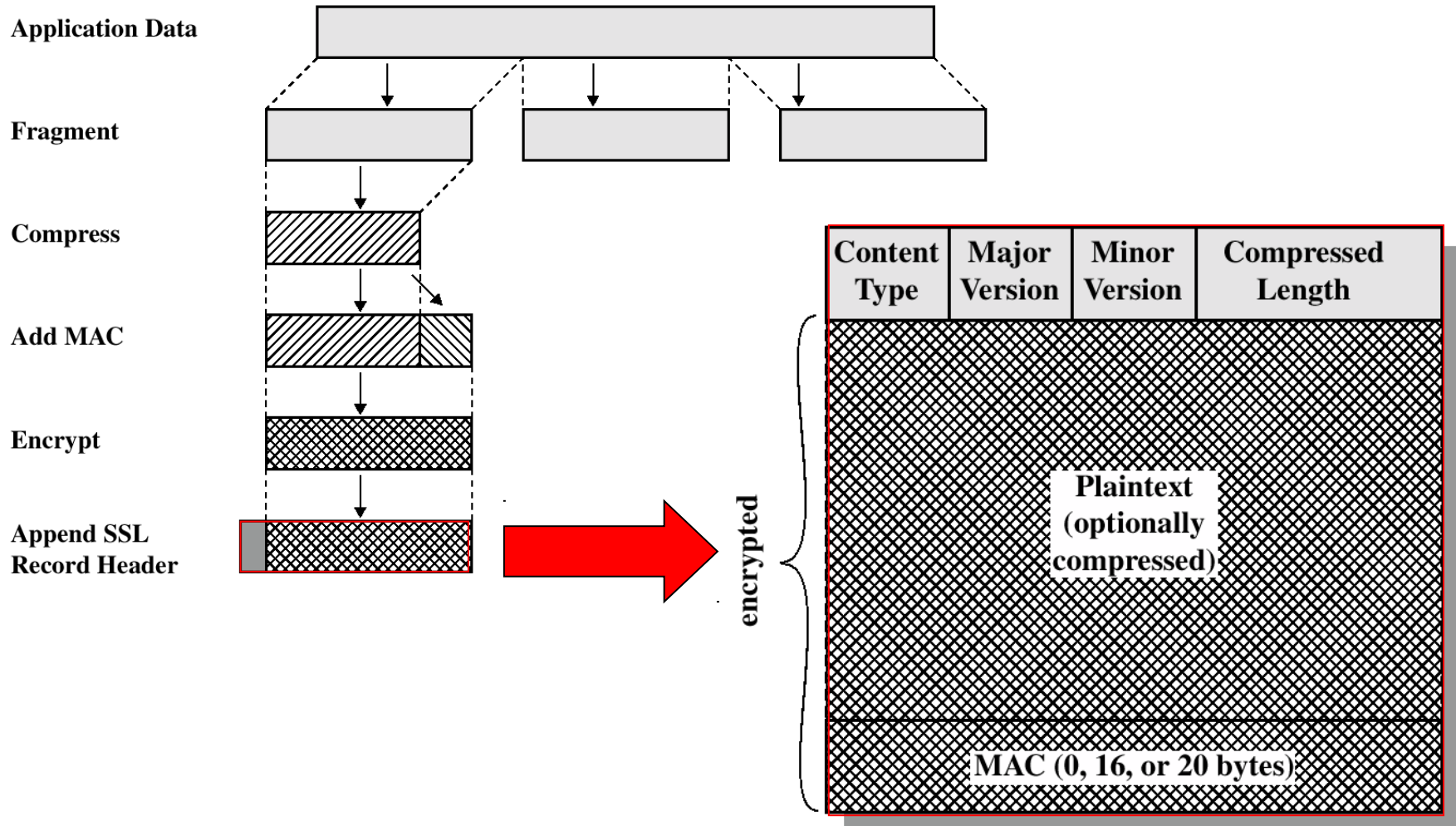
↑
tipo di contenuto
(change_cipher_spec, alert,
handshake, application_data)

↑
numero principale di versione

↑
numero secondario di versione

↑
lunghezza frammento
in chiaro (compressato)

SSL RP – formato del record



Transport Layer Security (TLS)

- Standard IETF, poche differenze con SSL3.0
- Stesso formato del record (lucido prec.)
- TLS1.0, Major Version = 3, Minor Version = 1
 - Se un server che non supporta TLS riceve una richiesta di connessione con numero di versione 3.1, l'handshake continua con SSL3.0
 - Analogamente, un server che supporta TLS risponderà ad una richiesta SSL3.0 con un handshake SSL3.0

TLS1.0 versus SSL3.0

■ Nuovi allarmi fatal

1. **decryption_failed**
A TLSCiphertext decrypted in an invalid way: either it wasn't an even multiple of the block length or its padding values, when checked, weren't correct.
2. **record_overflow**
A TLSCiphertext record was received which had a length more than $2^{14}+2048$ bytes, or a record decrypted to a TLSCompressed record with more than $2^{14}+1024$ bytes. This message is always fatal.
3. **unknown_ca**
A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or couldn't be matched with a known, trusted CA.
4. **access_denied**
A valid certificate was received, but when access control was applied, the sender decided not to proceed with negotiation.
5. **decode_error**
A message could not be decoded because some field was out of the specified range or the length of the message was incorrect.
6. **export_restriction**
A negotiation not in compliance with export restrictions was detected; for example, attempting to transfer a 1024 bit ephemeral RSA key for the RSA_EXPORT handshake method.
7. **protocol_version**
The protocol version the client has attempted to negotiate is recognized, but not supported. (For example, old protocol versions might be avoided for security reasons).
8. **insufficient_security**
Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.
9. **internal_error**
An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue (such as a memory allocation failure).

TLS1.0 versus SSL3.0

■ Nuovi allarmi warning

1. **decrypt_error**

A handshake cryptographic operation failed, including being unable to correctly verify a signature, decrypt a key exchange, or validate a finished message.

2. **user_canceled**

This handshake is being canceled for some reason unrelated to a protocol failure. If the user cancels an operation after the handshake is complete, just closing the connection by sending a close_notify is more appropriate. This alert should be followed by a close_notify.

3. **no_renegotiation**

Sent by the client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these would normally lead to renegotiation; when that is not appropriate, the recipient should respond with this alert; at that point, the original requester can decide whether to proceed with the connection. One case where this would be appropriate would be where a server has spawned a process to satisfy a request; the process might receive security parameters (key length, authentication, etc.) at startup and it might be difficult to communicate changes to these parameters after that point.

TLS1.0 versus SSL3.0

- Usa **HMAC** standard

- ☐ $\text{HMAC.H}(M) = \text{H}(K \oplus \text{pad2}, \text{H}(K \oplus \text{pad1}, M))$
- ☐ XOR anziché concatenazione (lucido 43)

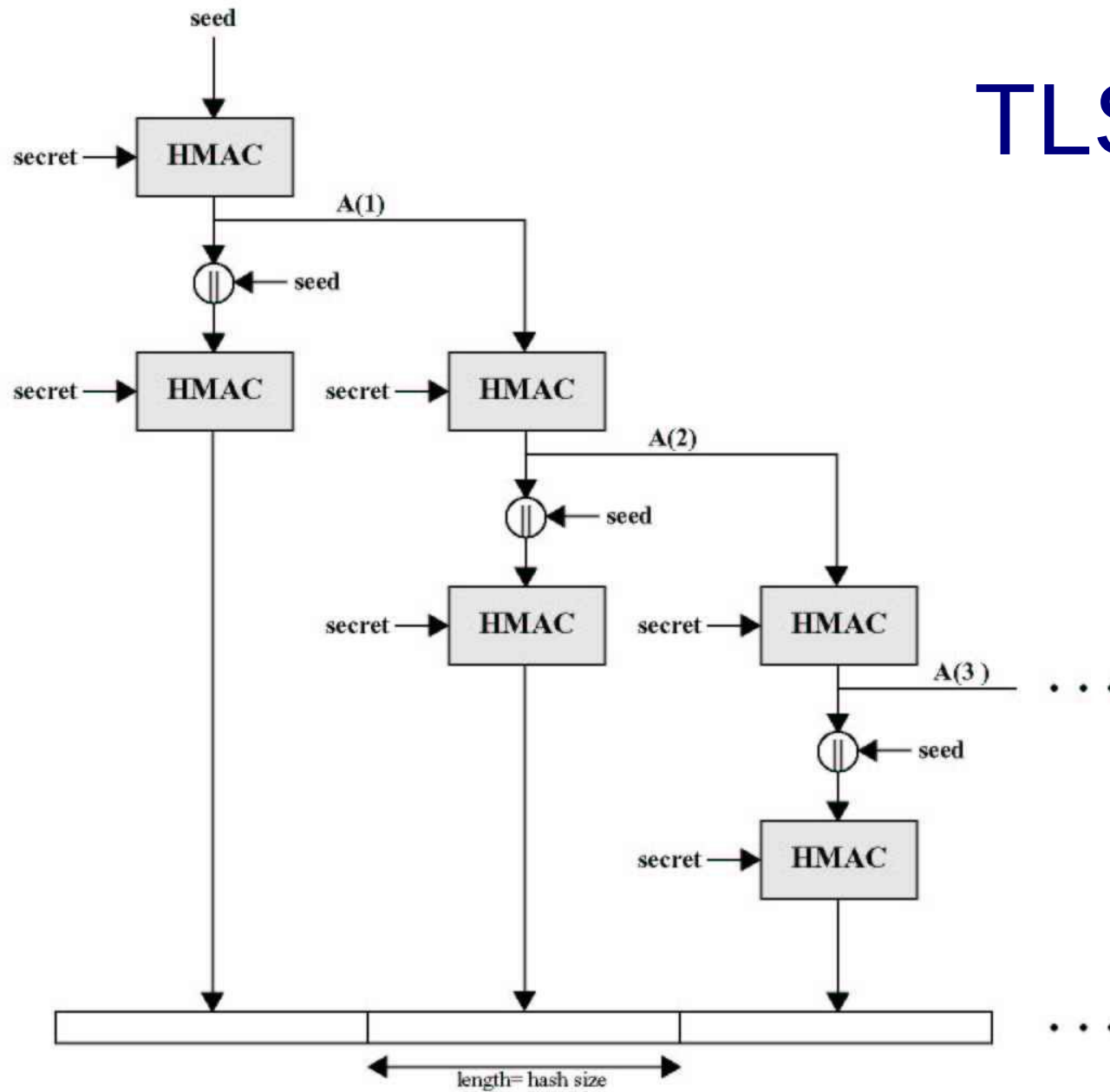
$\text{H}(\text{MAC_write_secret} \oplus \text{pad2},$
 $\text{H}(\text{MAC_write_secret} \oplus \text{pad1}, \text{seq_num},$
 $\text{TLSCompressed.type},$
 $\text{TLSCompressed.length},$
 $\text{TLSCompressed.fragment}))$

- Usa una **PseudoRandomFunction** (PRF) per espandere in maniera sicura segreti piccoli in segreti più lunghi

TLS: P.H (non è la PRF)

- MS usato come **seed** (seme)
- ClientHello.random e ServerHello.random usati come segreti per espandere il seme
 - $P.H(secret, seed) = \text{HMAC.H}(secret, A(1), seed), \text{HMAC.H}(secret, A(2), seed), \text{HMAC.H}(secret, A(3), seed), \dots$
 - $A(0) = seed, A(i) = \text{HMAC.H}(secret, A(i-1))$
 - 2 applicazioni di HMAC ad ogni iterazione

TLS: P.H



TLS: PRF

- La PRF usa due funzioni hash sicchè è sicura se almeno una delle due funzioni lo è
- Secret spezzato in due metà s_1 ed s_2
 - $\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P.MD5}(s_1, \text{label}, \text{seed}) \oplus \text{P.SHA-1}(s_2, \text{label}, \text{seed})$
 - Il secondo parametro di P.H è la concatenazione di label e seed

TLS1.0 versus SSL3.0

- `certificate_verify`: rimosso MS e padding
 - $H(H(\text{handshake_messages}))$
- `finished`: trasporta un hash diverso
 - $\text{PRF}(\text{MS}, \text{"finished"}, \text{MD5}(\text{handshake_messages}), \text{SHA-1}(\text{handshake_messages}))$
 - Il terzo parametro di PRF è la concatenazione dei due hash

TLS1.0 versus SSL3.0

- MS calcolato diversamente
 - $MS1 = \text{PRF}(\text{PMS}, \text{"master_secret"}, \text{ClientHello.random}, \text{ServerHello.random})$
 - $MS2 = \text{PRF}(MS1, \text{"master_secret"}, \text{ClientHello.random}, \text{ServerHello.random})$
 - ...
 - $MS = MS1, MS2, \dots$
- Algoritmo iterato fino ad ottenere 48 byte

TLS1.0 versus SSL3.0

- I bit per le varie chiavi ottenuti con algoritmo analogo ma non identico
 - $\text{key_block1} = \text{PRF}(\text{MS}, \text{"key_expansion"}, \text{ClientHello.random}, \text{ServerHello.random})$
 - $\text{key_block2} = \text{PRF}(\text{key_block1}, \text{"key_expansion"}, \text{ClientHello.random}, \text{ServerHello.random})$
 - ...

Concludendo

- SSL/TLS danno buoni risultati
- Sono universalmente accettati ma...
- ... danno molto “meno” di SET di Visa/Mastercard, il quale
 - ☐ Gestisce la registrazione con una CA
 - ☐ Gestisce anche la fase di pagamento
 - ☐ Permette **condivisione parziale** di informazioni
 - ☐ Tuttavia non ha preso piede