La Sicurezza nei Sistemi Distribuiti

Cosa proteggere e da chi?

Risorse e Informazioni

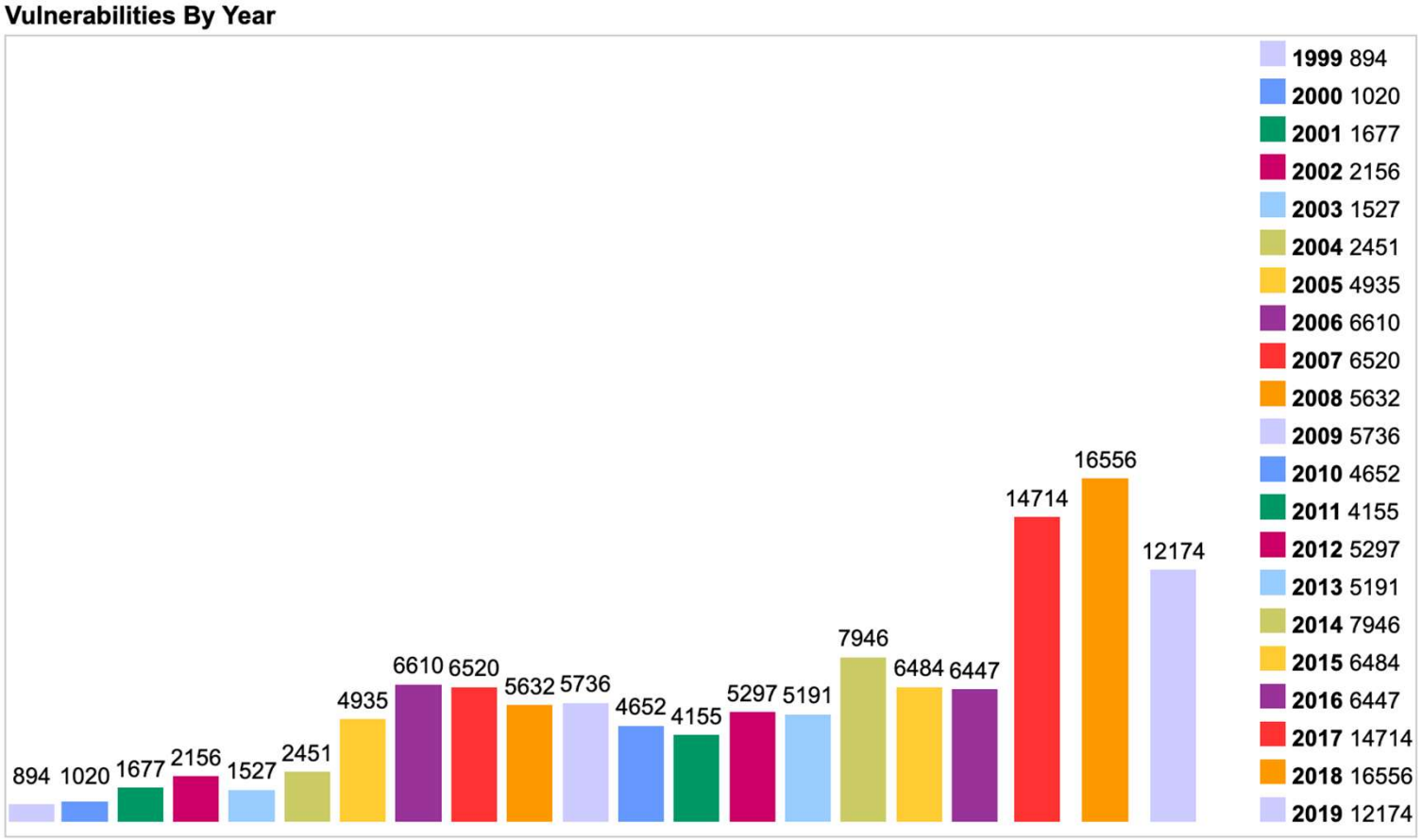
Risorse: esecuzione, memorizzazione, comunicazione

Negli ultimi anni, il numero di attacchi alla sicurezza in Internet è continuamente aumentato, ma in proporzione di meno dello sviluppo della rete. Questo per l'accresciuta sensibilità verso i temi della sicurezza. Dati forniti dal CERT (www.cert.org).

Con il termine CERT (Computer Emergency Response Team) si identificano dei centri di studio indipendenti tra loro (legati a Università ed enti di ricerca) su vari temi legati alla sicurezza attivi in moltissimi paesi. Tipicamente ricevono segnalazioni di incidenti e violazioni in Internet, pubblicano i problemi rilevati e le relative soluzioni, coordinano le azioni di moltissimi esperti per fronteggiare i problemi di sicurezza nella rete.

Statistiche CVE

(Common Vulnerabilities and Exposures)



Fonte: http://www.cvedetails.com/browse-by-date.php (dato del 9/12/2020)

Aspetti del problema Sicurezza

La sicurezza ha moltissimi aspetti:

Autenticazione	*Authentication*
Autorizzazione	*Authorisation*

Riservatezza	*Privacy*
Disponibilità	*Availability*
Integrità	*Integrity*
Paternità	*Non-repudiability*

Autenticazione

Verifica dell'identità dell'utente attraverso:

- Possesso di un oggetto (es., smart card)
- Conoscenza di un segreto (password)
- Caratteristica personale fisiologica (impronta digitale, venature retina)

Problema della mutua autenticazione

Attenzione!!! **Autenticazione ≠ Autorizzazione**

Autorizzazione

Specifica le azioni concesse a ogni utente

Riservatezza

Previene la lettura non autorizzata delle informazioni. (es. un intruso che intercetta un messaggio non è in grado di interpretarlo, di risalire al contenuto informativo del msg)

Integrità

Previene la modifica non autorizzata delle informazioni. (es. un messaggio spedito dal mittente è ricevuto tale e quale dal destinatario)

Alcune modifiche dell'informazione possono non alterare l'integrità

Disponibilità

Garantire in qualunque momento la possibilità di usare le risorse

Paternità

Chi esegue un'azione non può negarne la paternità, per esempio un mittente non può ripudiare di aver scritto un certo messaggio

Esempio

Alice manda messaggio a **Bob:** costruiscimi una casa per 1 milione

Problemi

1. **Intrusione**: **Trudy** intercetta il messaggio e cambia 1 milione in 100 milioni (violata l'integrità del messaggio)
   Soluzione: cifro il messaggio, Trudy può intercettare il messaggio ma non lo sa leggere (preservata riservatezza).

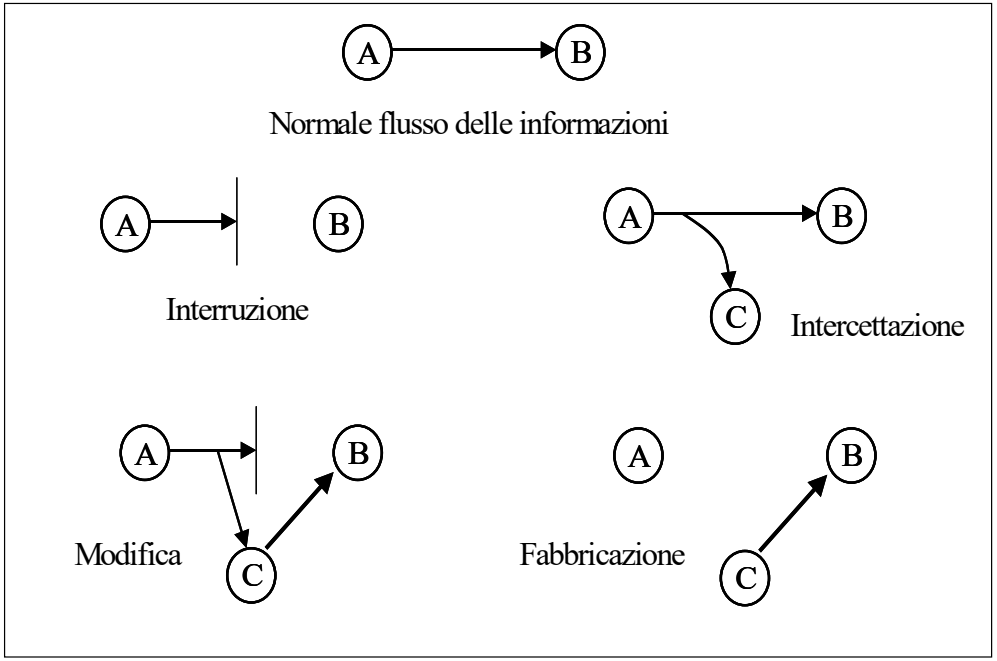
2. **Non fiducia tra Alice e Bob**: il messaggio è cifrato ma sia Bob sia Alice conoscono come cifrare il messaggio e possono mentire, senza che l'altro possa dimostrarlo. Due casi:
   - Alice non ha più i soldi, ma Bob ha costruito la casa
   - Bob costruisce una casa di costo 10 milioni

   In regime di mutuo sospetto nessuno può provare a una terza parte che l'altro sta mentendo. **Soluzione**: Alice firma il messaggio, chiunque (per es. un giudice) potrà verificare che Alice ha chiesto una casa da 1 milione e che così è scritto nel messaggio

Attacchi alla sicurezza di sistema: tipi di attacco

Tentativi passivi

- leggere informazioni di altri
- eseguire analisi del sistema e del traffico



Tentativi attivi

- modifica dell'informazione (es., messaggio, file ...)
- cancellazione dell'informazione (es., messaggio, file ...)
- impersonare un altro utente
- accesso non autorizzato alle risorse
- rendere un sistema impossibilitato a fornire i servizi

Criteri per valutare la sicurezza dei sistemi

Orange Book
redatto in ambito militare (U.S.) si concentra sulla riservatezza dei dati (non sull'integrità) e sul mandatory access control

Red Book (TNI)
Applica i criteri di sicurezza definiti nell'Orange Book ai sistemi interconnessi in rete

Lavender Book (TDI)
Applica i criteri di sicurezza definiti nell'Orange Book ai data base management system

Orange Book, Red Book e Lavender Book costituiscono la "Rainbow" series

Criteri per valutare la sicurezza dei sistemi

Information Technology Security Evaluation Criteria (ITSEC) (Germania, Francia, UK, Olanda) è un insieme di criteri per valutare la sicurezza dei sistemi informatici (anni ‘90).

ITSEC è stato sostituito dal Common Criteria (CC) sviluppato in modo congiunto da USA, Canada ed Europa.

CC è lo standard internazionale attualmente più avanzato per la certificazione della sicurezza dei sistemi informatici.

Esempio: Red Hat Enterprise Linux è certificato EAL4+ del CC

Critiche: tutti i processi di verifica della sicurezza (CC od Orange book e ITSEC) sono molto costosi (> 100k $) e impiegano molto tempo. Spesso esaminano solo la documentazione fornita, non il codice del sistema.

Orange Book

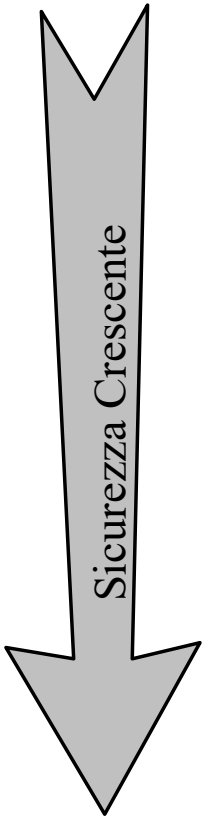
Ogni livello ingloba gli elementi di sicurezza dei precedenti e vi aggiunge quelli caratteristici del livello stesso

Categoria D: **non sicuri**

Categoria C: **sistemi discrezionali**
- Protezione oggetti a discrezione dell'utente
- il sistema non forza criteri
- assunzione di base: buonafede utenti
- (protezione dall'esterno)

Categoria B: **sistemi non discrezionali, protezione obbligatoria**
- Oggetti protetti in modo obbligatorio dal sistema
- assunzione di base: possibile malafede utenti

Categoria A: **modello formale di sicurezza; informazioni classificate**



Orange Book

Categoria D: **non sicuri**

Categoria C: **protezione possibile**

classe C1:

- autenticazione utenti via password
- semplici controlli degli accessi alle risorse (owner/group/other)
- accesso controllato degli utenti a certe parti della memoria

classe C2:

- controllo degli accessi alle risorse con granularità utente
- cancellazione memoria prima del suo assegnamento a utenti
- auditing

Orange Book

Categoria B: **protezione obbligatoria**

Questa categoria di sistemi è basata su alcune considerazioni specifiche (protezione obbligatoria):

1. La protezione di un oggetto non è decisa dal proprietario dell'oggetto
2. Il sistema assicura e impone la protezione degli oggetti

Si utilizzano etichette di sicurezza:

- soggetti= processi
- oggetti = file (regular file, directory, device, etc.), processi

In ogni momento tutti i soggetti e gli oggetti del sistema hanno un'etichetta (unclassified, confidential, secret, top secret).

Orange Book

Restrizioni nell'accesso ai file:

- **Lettura** file consentita solo se l'etichetta del soggetto lettore è maggiore dell'etichetta del file (no read-up)
- **Scrittura** di un file consentita se l'etichetta del file è MAGGIORE dell'etichetta del soggetto (no write-down)

Esempio:

- Processo (soggetto) Secret non può leggere file etichettato Top Secret
- Processo (soggetto) Secret non può scrivere un file Confidential

Orange Book

Categoria B: **protezione obbligatoria**

classe B1:

- etichette di sicurezza attaccate a tutti gli utenti, i processi e le risorse, S.O. controlla i read-up e i write-down
- tutti i dispositivi (stampanti etc.) devono trattare opportunamente le etichette di sicurezza

classe B2:

- trusted path tra S.O. e utente, presenza cioè di meccanismi per garantire che un utente parli al S.O. (trojan horse)
- processi che cambiano livello di sicurezza notificano utente
- strutturazione del kernel: concentrazione delle funzionalità di sicurezza all'interno di un piccolo "security kernel"
- identificazione dei covert channel e delle loro bande di trasmissione
- strette procedure di controllo nella modifica delle parti del sistema sensibili per la sicurezza

Orange Book

classe B3:

- controllo accessi per utenti con overrule
- auditing attivo
- secure crashing e restarting

Categoria A: **modello formale di sicurezza; informazioni classificate**

classe A1:

- stessi requisiti precedenti ma è richiesta la verifica del progetto

POSIX P1003.6

POSIX P1003.6 Aree funzionali indirizzate: auditing, discretionary access control, mandatory access control, information labels, privilege.

POSIX definisce le **interfacce** necessarie alla gestione delle informazioni di sicurezza, per realizzare applicazioni portabili.

P1003.6 migliora meccanismi di sicurezza di P1003.1. Esempio:

- P1003.1 adotta il meccanismo dei **permission bit**:
  - ☺semplice ed economico
  - ☹scarsa granularità nel controllo degli accessi (non si possono indirizzare utenti/gruppi specifici)
- P1003.6 adotta le **Access Control List** (ogni oggetto ha una ACL associata che specifica i diritti di accesso sulla risorsa per gruppi e utenti)

NB. POSIX NON specifica come implementare le ACL né la rappresentazione interna delle ACL.

Discretional Access Control in POSIX P1003.6

I diritti di accesso a una risorsa sono decisi dal proprietario della risorsa stessa.

Mandatory Access Control in POSIX P1003.6

Uso di un labeling mechanism. Tutti i soggetti e oggetti del sistema hanno una MAC label, in ogni momento (unclassified, confidential, secret, top secret).

Restrizioni nell'accesso ai file:

- **Lettura** file consentita se la label del processo lettore è maggiore della label del file (no read-up)
- **Scrittura** di un file consentita se la label del file è MAGGIORE della label del processo (no write-down).

Privilegi in POSIX P1003.6

Il **meccanismo dei privilegi** controlla i diritti di accesso alle risorse da parte degli utenti e dei gruppi. Può fornire a specifici utenti la capacità di eseguire operazioni “sensibili” ai fini della sicurezza, sotto certe condizioni e per un tempo limitato.

Esempio. Per eseguire un **backup** del sistema, l’amministratore di sistema deve acquisire il **privilegio di lettura** su tutti i file (**override** delle informazioni di controllo dell’accesso su tutti i file)

Esempio. UNIX super user è un meccanismo di overriding dei privilegi di tipo “tutto o niente”

P1003.6 suggerisce un meccanismo di tipo “**least privilege**”, per permettere un override del minor numero possibile di informazioni di sicurezza per lo svolgimento di un determinato compito.

Caratteristiche di un meccanismo di overriding dei privilegi: granularità e durata temporale

OWASP

La Fondazione OWASP è stata istituita come organizzazione internazionale non-profit nell'Aprile 2004.

Tutti gli strumenti, i documenti, i forum e i capitoli di OWASP sono open source a chiunque sia interessato a migliorare la sicurezza delle applicazioni. Si veda https://www.owasp.org

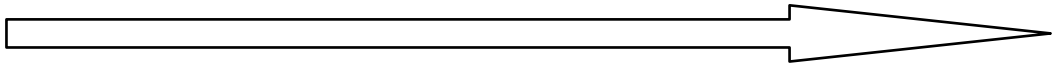
OWASP sostiene l'approccio alla sicurezza delle applicazioni come problema:

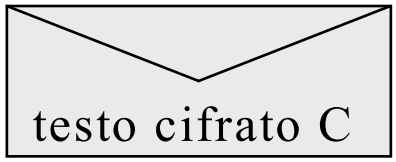
- personale;
- di processo;
- tecnologico.

Questo poiché gli approcci più efficaci alla sicurezza delle applicazioni non sono a senso unico e includono miglioramenti in tutte le aree di sviluppo.

Crittografia

la scienza della crittografia fornisce gli strumenti per rendere un messaggio non intelligibile per chiunque non ne sia il destinatario

$E(P) = C$  $P = D(C)$



E = funzione di cifratura

D = funzione di decifrazione

P = plaintext, testo in chiaro

C = ciphertext, testo cifrato

La crittografia è una scienza che ha radici antiche. Rilevanti testimonianze storiche di importanti esempi di crittografia di sostituzione e di trasposizione.

Codici di sostituzione (monoalfabetica)

Una permutazione sull'alfabeto del testo chiaro

Se ogni simbolo corrisponde a un byte alfabeto di 256 simboli
==>

ogni **simbolo** si trasforma in un altro simbolo dell'alfabeto.

Attenzione: cambiano simboli ma non cambia il loro ordine

256! possibili **permutazioni** dell'alfabeto

La chiave di cifratura è la tabella con le corrispondenze di tutti i simboli (tabella 256x2)

Codici di sostituzione: Cifrario di Cesare

Cifrario di Cesare (riferito nel De Bello Gallico) è un codice di sostituzione monoalfabetica in cui ogni carattere viene sostituito con quello che lo segue di 3 posizioni nell'ordinamento alfabetico (la chiave in questo caso è 3).

Cifrario di Cesare

regola di sostituzione: A<=>D, B<=>E, C<=>F, D<=>G....Z<=>C

esempio di testo in chiaro: de bello gallico

esempio di testo cifrato: gh ehoor ldoonfr

Facilmente attaccabile

Codici di sostituzione polialfabetici (Leon Battista Alberti, Enigma,…)

Codici di trasposizione

Si fissa un numero intero P (periodo della trasposizione) e si sceglie una permutazione degli interi da 1 a P

se P=7 si può fare corrispondere alla successione

1 2 3 4 5 6 7 quella permutata 4 6 3 5 7 1 2

Attenzione: i simboli dell'alfabeto sono gli stessi ma cambia l'ordine

Il testo in chiaro viene considerato a **blocchi di lunghezza P** byte e i **P byte di ciascun blocco** vengono anagrammati in base alla permutazione. Per la decifrazione si usa la permutazione inversa

Politiche miste

cifrario semplice: unica trasformazione elementare

cifrario composto o di prodotto: una serie di trasformazioni elementari in cascata

Crittoanalisi (Analisi crittografica)

La crittografia può essere attaccata, per decifrare un messaggio anche senza conoscere la chiave. Si possono avere due tipologie di approccio:

- **Analisi crittografica.** Il crittoanalista cerca di decifrare studiando il sistema di crittografia e avendo a disposizione della conoscenza.
- **Attacco di forza bruta.** Si provano tutte le possibili chiavi di decifrazione.

Attacco	Conoscenza a disposizione per l'analisi crittografica
con solo testo cifrato (ciphertext only)	Testo cifrato da decifrare Proprietà statistiche del linguaggio
con testo in chiaro noto (known plaintext)	Testo cifrato da decifrare Termini presenti nel testo in chiaro e corrispondenti testi cifrati (per es. successivamente pubblicati) (coppie <plaintext, ciphertext>)
con testo in chiaro scelto (chosen plaintext)	Testo cifrato da decifrare Testi cifrati corrispondenti a qualsiasi testo in chiaro che si ritenga possa essere utile ai fini della crittanalisi (coppie <scelto plaintext, ciphertext>)
con testo cifrato scelto (chosen ciphertext)	Qualsiasi testo cifrato che si ritenga possa essere utile ai fini della crittanalisi e il corrispondente testo in chiaro (coppie <plaintext, scelto ciphertext>)

La crittografia moderna

La crittografia moderna usa due tipi di algoritmi:

- a chiave segreta (simmetrici)
- a chiave pubblica (asimmetrici)
- omomorfica (in fase di sviluppo)

Un sistema di crittografia comprende:

- **algoritmo**
- **chiave**

SICUREZZA DI UN SISTEMA DI CRITTOGRAFIA DIPENDE DA:

- segretezza della chiave
- robustezza dell'algoritmo
- segretezza dell'algoritmo ?

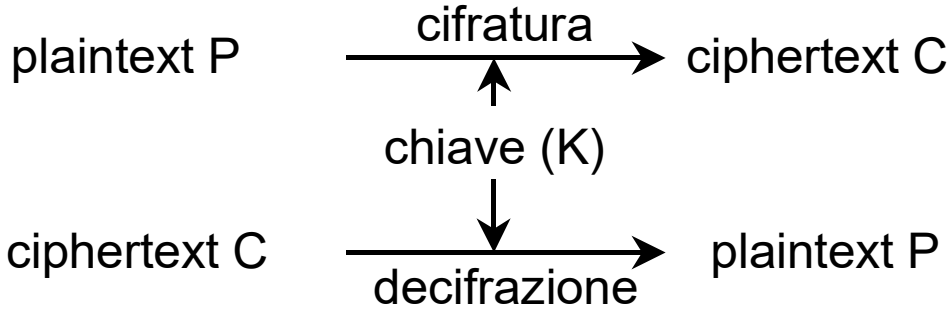
La crittografia moderna

Gli algoritmi di cifratura sono resi noti in ambiti “civili”, mentre sono mantenuti segreti in ambiti militari.

In ogni caso, nessun sistema è *assolutamente* sicuro. Un sistema di crittografia è **computazionalmente sicuro** se:

- il costo per la crittoanalisi è superiore al valore dell’informazione
- il tempo richiesto per la crittoanalisi è superiore al tempo di vita dell’informazione

La crittografia a chiave segreta (simmetrica)



funzione di cifratura C = E(P, K) = EK (P)

funzione di decifrazione P = D (C, K) = DK (C)

Con funzioni di cifratura e decifrazione l'una inversa dell'altra

EK (DK (C)) == C

DK (EK (P)) == P

Sistemi con una chiave unica per cifrare e decifrare (**K**) o con due chiavi diverse per cifrare e decifrare, ma comunque tra di loro collegate ($\mathbf{K_e}$, $\mathbf{K_d}$)

Impiego dei sistemi a chiave segreta

Trasmissione su un canale insicuro

il messaggio cifrato non rivela l'informazione anche se intercettato

richiede: condivisione chiave tra mittente e destinatario

Immagazzinamento sicuro su un media insicuro

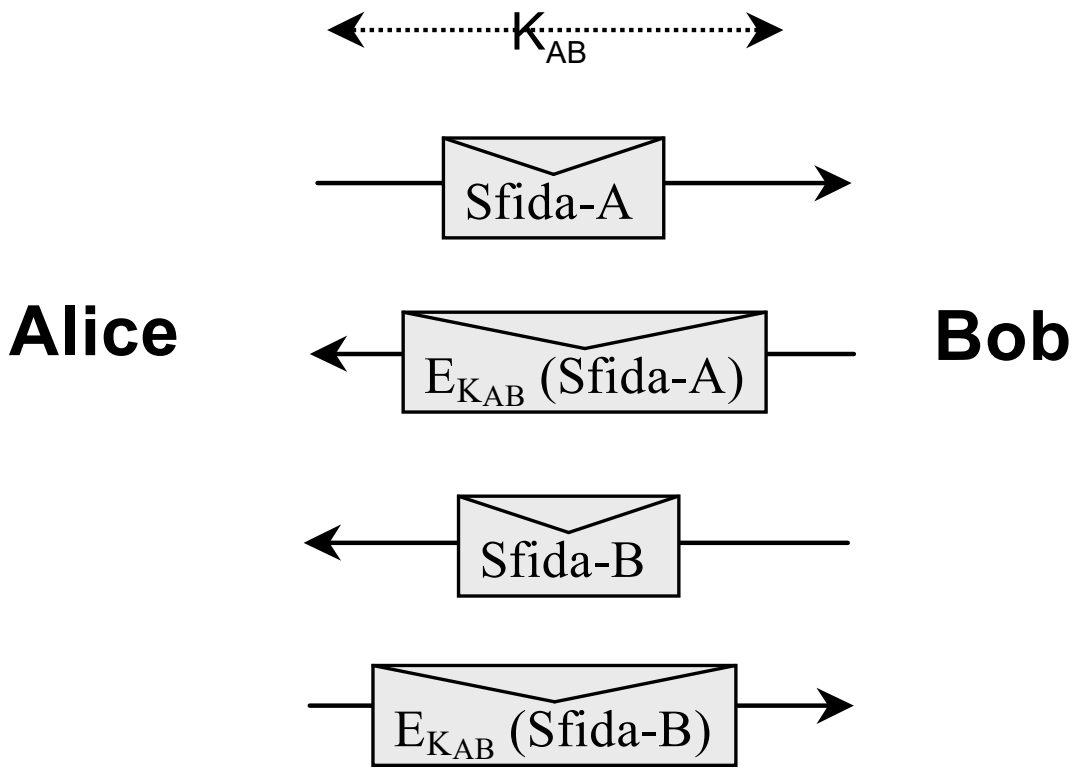
es. cifratura del file system

Integrità dei messaggi

Message Integrity Code (MIC): il checksum cifrato del messaggio

Impiego dei sistemi a chiave segreta

Autenticazione (e mutua autenticazione)

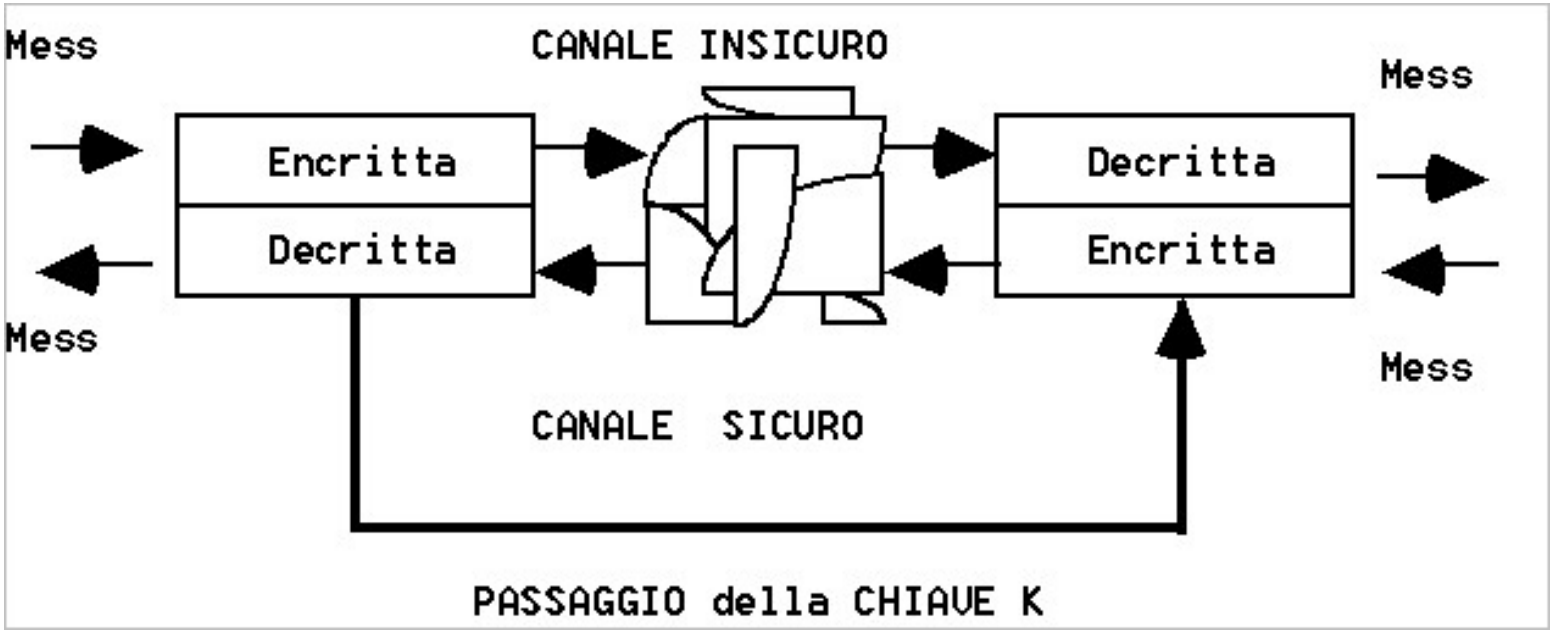


Sistemi a chiave segreta

Algoritmo di pubblico dominio per la cifratura

IBM Data Encryption Standard (DES)

DES (1977) *cifrario composto o di prodotto* con chiave di 56 bit con blocchi di 64 bit. *Algoritmo in 19 stage.*



Sistemi a chiave segreta

Critiche al DES:

insufficiente la lunghezza della chiave (56 bit)

nel progetto originale pare fosse maggiore (128 bit)

Altri algoritmi standard pubblicati da National Institute of Standards and Technology (NIST):

- TripleDES (3DES), con chiavi di 168 bit (3x56)
- AES (Advanced Encryption Standard) (2001) con chiavi di varia lunghezza, 128 o 192 o 256 bit

Altri sistemi a chiave segreta, con diversi algoritmi e diverse lunghezze di chiave:

Serpent (più sicuro di AES ma 3 volte più lento), Twofish, …

Cifratura a blocchi

I sistemi a chiave segreta sono pensati per essere usati in cifrari a blocchi al fine di cifrare un flusso di informazioni in tempo reale

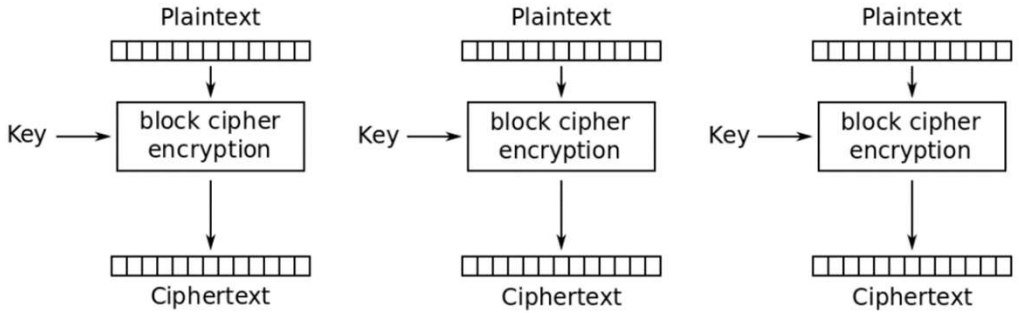
Diverse possibili modalità:

- ECB
- OFB
- CBC
- CTR

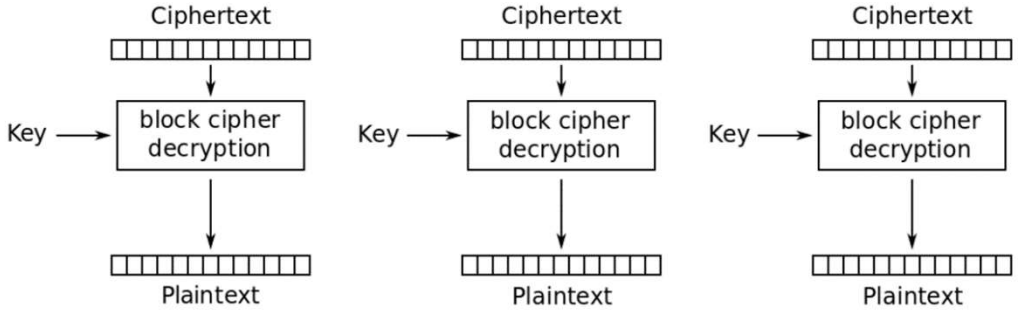
(e varie sotto-modalità) con diversi pro e contro.

Al momento gli esperti ritengono che le modalità CBC con random IV e CTR siano le più sicure.

Electronic Codebook (ECB)



Electronic Codebook (ECB) mode encryption

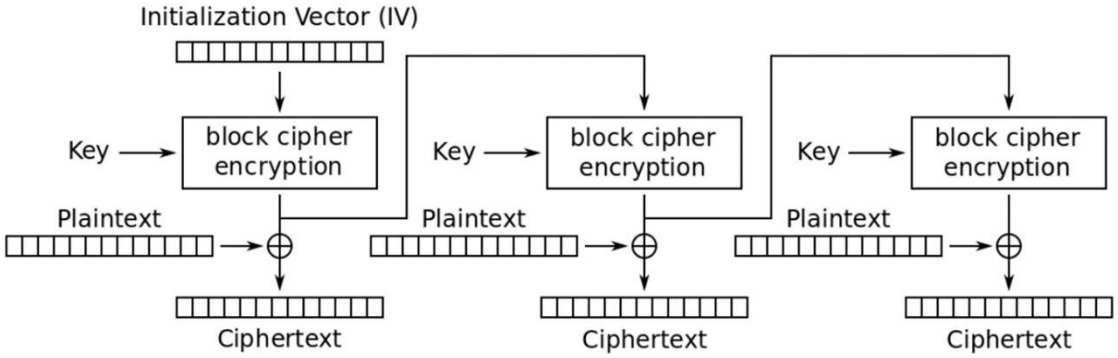


Electronic Codebook (ECB) mode decryption

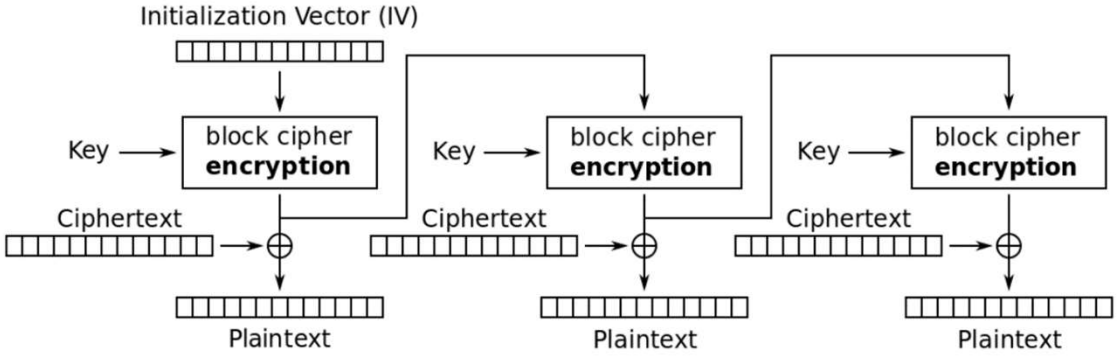
Soluzione naive, che produce ciphertext identici in risposta a plaintext identici

- Bassa entropia
- Suscettibilità a replay attacks

Output Feedback (OFB)



Output Feedback (OFB) mode encryption

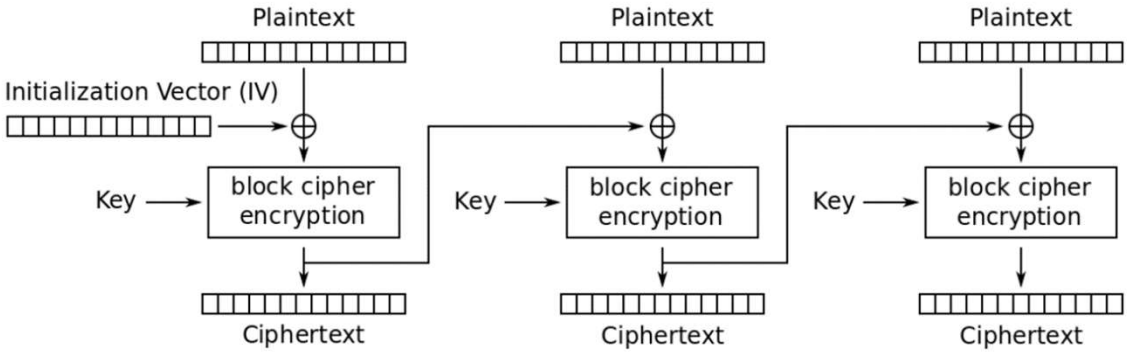


Output Feedback (OFB) mode decryption

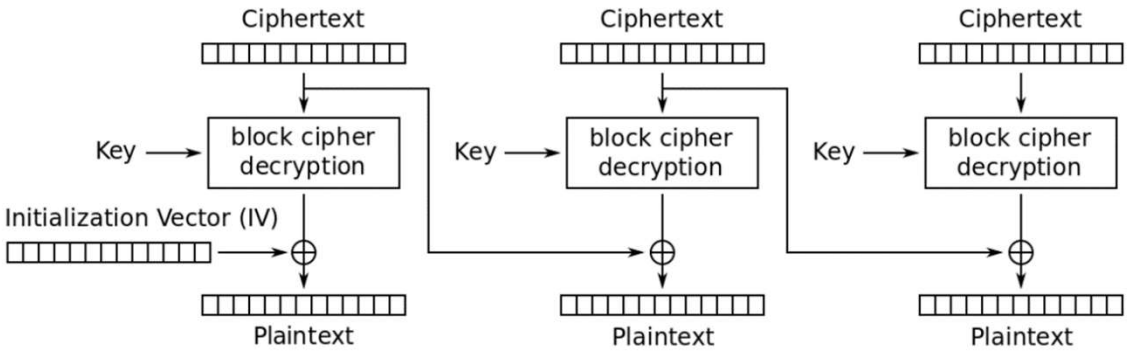
Soluzione leggermente meno naive, che però presenta un grosso problema: se si cifra lo stesso messaggio con uno stesso IV si riusa la stessa sequenza di chiavi

- Suscettibilità a known plaintext attack

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



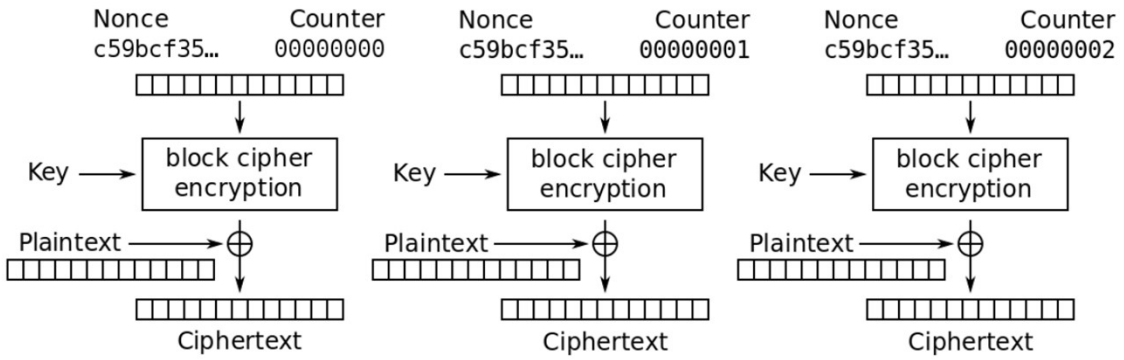
Cipher Block Chaining (CBC) mode decryption

Soluzione molto utilizzata, ma non priva di problemi:

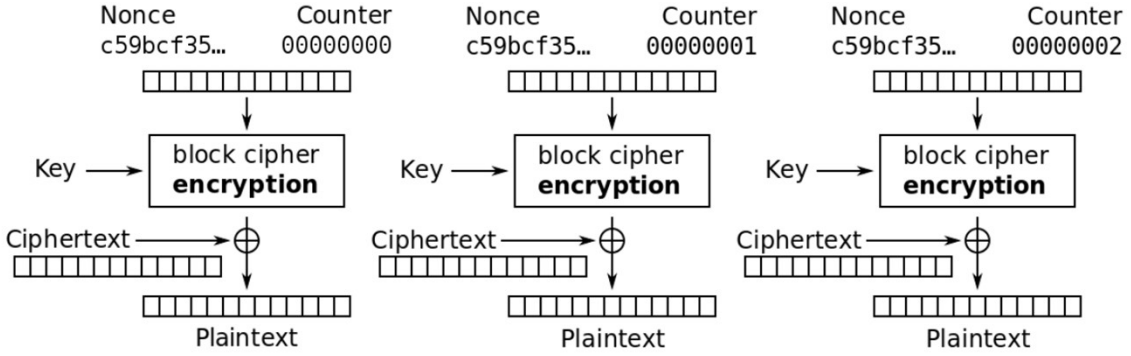
- Necessità di padding
- Operazioni non parallelizzabili

Generalmente considerata sicura se usata con random IV

Counter (CTR)



Counter (CTR) mode encryption

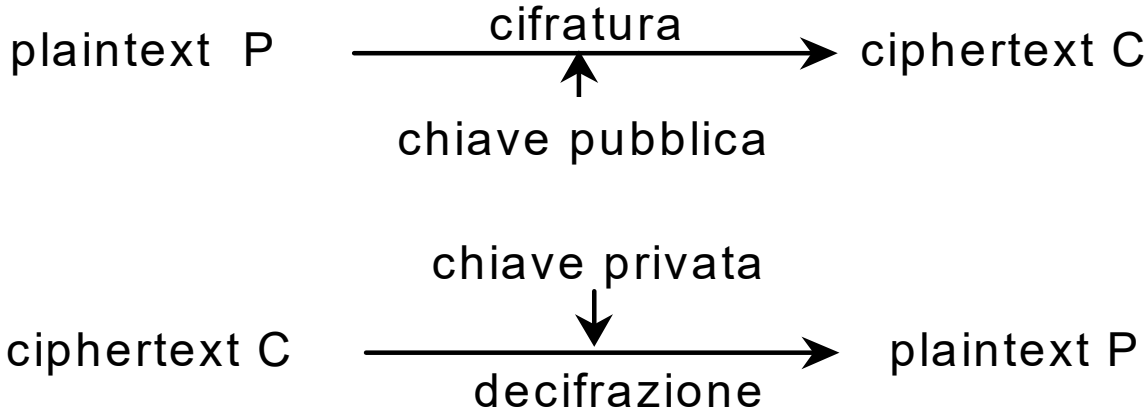


Counter (CTR) mode decryption

Soluzione robusta e di semplice utilizzo, ma sicura solo in caso di garanzia di unicità della coppia (chiave, nonce) – che quindi non può essere riutilizzata!

La crittografia a chiave pubblica (asimmetrica)

Usa una coppia di chiavi (chiave pubblica e chiave privata) per cifrare/decifrare. È molto difficile dedurre da una chiave il valore dell'altra.



funzione di cifratura C = E(P, Kpub) = Ekpub (P)

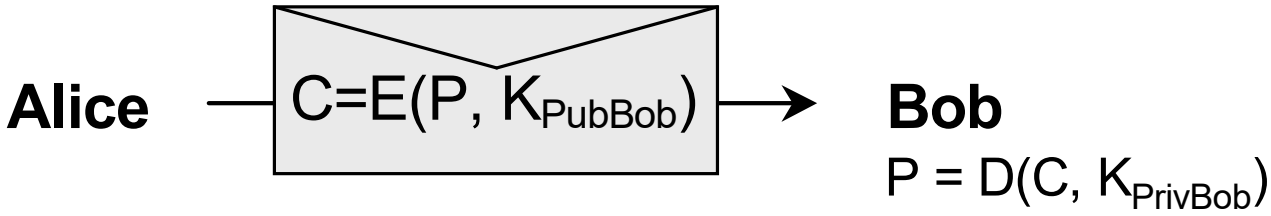
funzione di decifrazione P = D (C, Kpriv) = Dkpriv (C)

La chiave pubblica è resa di dominio pubblico.

Costi computazionali per cifratura e decifrazione molto superiori al caso simmetrico.

Impiego dei sistemi a chiave pubblica (asimmetrica)

Trasmissione su un canale insicuro: messaggio cifrato con la chiave pubblica del ricevente.

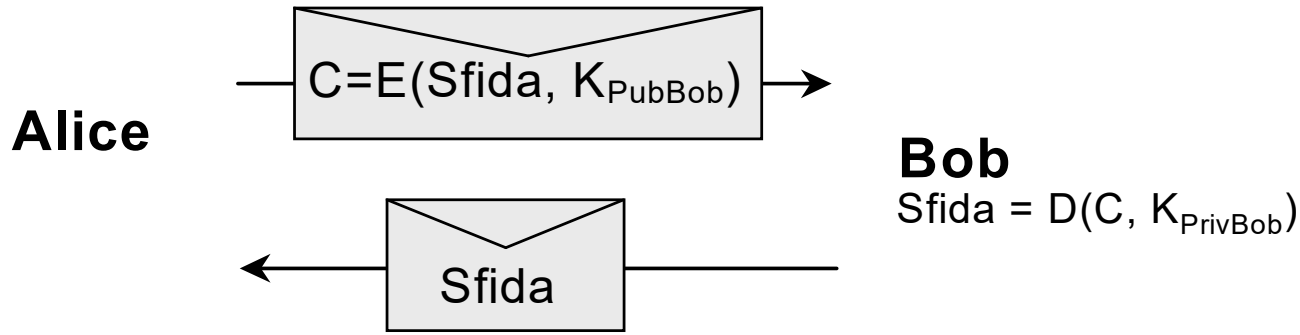


Solo Bob può decifrare C, ma non può sapere con certezza di chi sia il messaggio.

Immagazzinamento sicuro su un media insicuro: i dati vengono cifrati con la propria chiave pubblica.

Impiego dei sistemi a chiave pubblica (asimmetrica)

Autenticazione



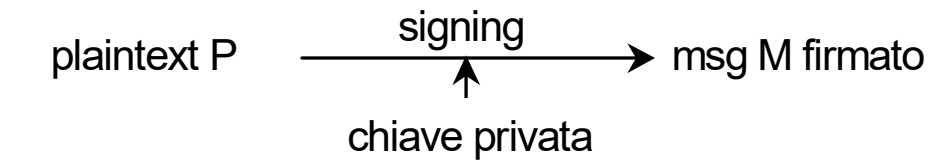
Solo Bob può decifrare C. Anche schemi per mutua autenticazione.

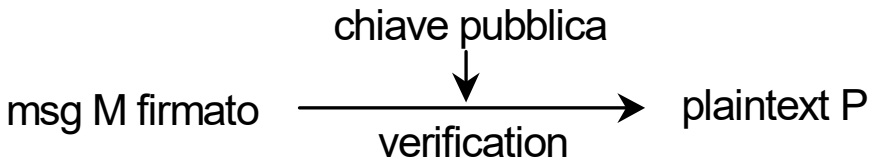
Questi problemi sono risolti anche con la crittografia a chiave segreta, ma nel caso di chiave pubblica:

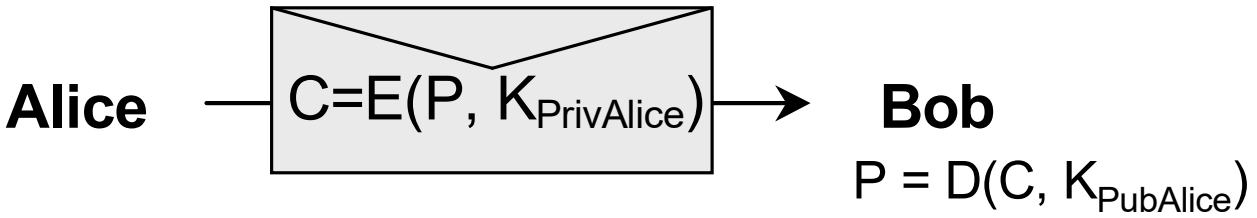
- ☺ non è richiesta condivisione chiave
- ☹ costo computazionale molto superiore

Impiego dei sistemi a chiave pubblica

Impiego principali di tali sistema per la firma digitale:



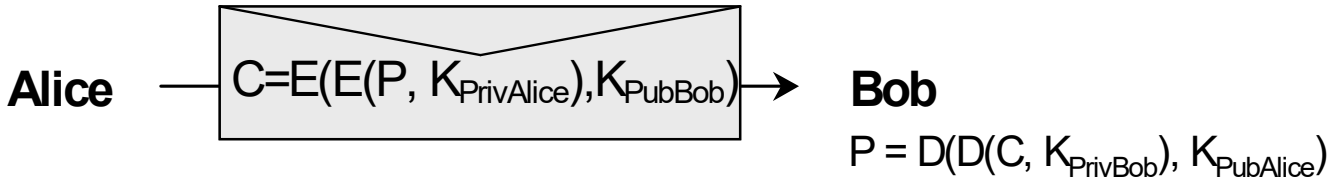




Chiunque riconosce la *paternità* di P (ad Alice), perché solo Alice conosce la propria chiave privata.

Impiego dei sistemi a chiave pubblica

Caso Composto (paternità e privacy)



Doppia cifratura:

Alice cifra P con la propria chiave privata poi cifra con la chiave pubblica di Bob.

Solo Bob può leggerlo e provare che arriva da Alice.

Sistemi a chiave pubblica

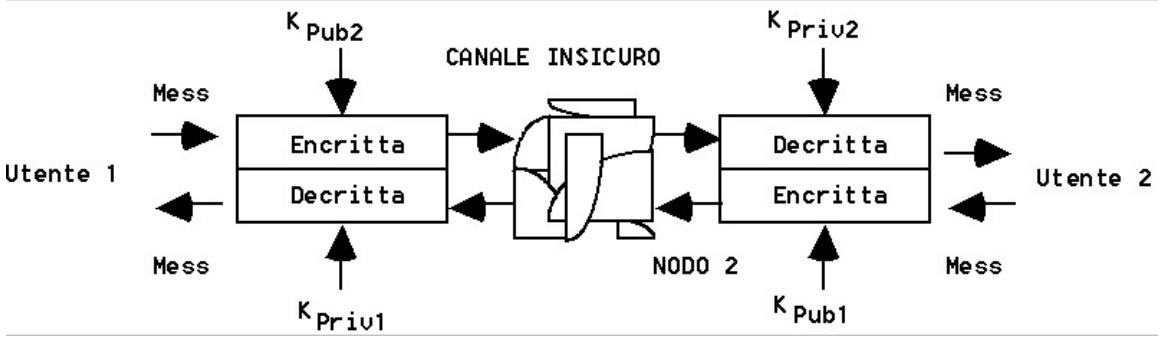
(Diffie Hellman 1976)

Cifratura separata dalla decifrazione, con coppia di chiavi diverse

Chiave pubblica **K_{Pub}**

e privata **K_{Priv}**

Invertibilità non facile. *Con chiave pubblica e algoritmo non si riesce analiticamente a ricavare la chiave privata*



Funzione unidirezionale:

una funzione facilmente computabile ma la cui inversa non può essere computata a meno di informazioni sulla sua costruzione

Algoritmo RSA (Rivest, Shamir, Adelman)

Uso di prodotto di **due numeri primi molto grandi p e q** ($> 10^{100}$)

N=p*q **Z= (p-1)*(q-1)**

d è un numero primo rispetto a Z

i i * d= 1 (mod Z)

i è quindi il più piccolo elemento della serie Z+1, 2Z+1, 3Z+1, ... divisibile per d

Il messaggio viene preso a blocchi di **f** bit, con $\mathbf{2^f} < N$

(M per blocco di messaggio e C per forma cifrata)

$\mathbf{K_{pub}}$ **== (N, i)** cifratura ➔ $M= C^i \bmod N$

$\mathbf{K_{priv}}$ **== (N, d)** decifrazione ➔ $C= M^d \bmod N$

e le due chiavi sono inverse

$\mathbf{K_{pub}}$ è nota (N, i) e $\mathbf{K_{priv}}$ (N, d) è la chiave privata

Ma attenzione all'implementazione del protocollo RSA!!!

http://book.huihoo.com/pdf/security-engineering/Papers/psandqs.pdf

Problemi in sistemi con algoritmi asimmetrici

E se il possessore delle chiavi se ne va:

il gestore deve conoscere entrambe le chiavi?

Attenzione: la difficoltà nella definizione delle coppie di chiavi pubblica/privata è la ricerca dei due numeri primi (p, q) molto grandi (che però è fatta una tantum)

fattorizzazione di un numero di 200 cifre richiede 4 miliardi di anni

l'operazione per un numero di 500 cifre 10^25 anni

(comune workstation, dati 1995)

Chiave pubblica o chiave segreta?

I sistemi di crittografia a chiave pubblica e quelli a chiave segreta presentano delle differenze significative che determinano il loro impiego in diversi contesti e con diversi modi.

Le differenze riguardano:

- diverso **costo computazionale**, che determina quindi la velocità di cifratura/decifrazione dei messaggi
- differente facilità di **gestione e distribuzione delle chiavi** tra gli utenti del sistema.

Costo computazionale: le prestazioni di DES ed RSA

Algoritmi a chiave pubblica (es., RSA) hanno un **costo computazionale molto superiore** al caso di chiave segreta (es., DES). La cifratura/decifrazione di un messaggio richiede tempi molto maggiori.

Realizzazioni Hw di RSA sono circa 1000 volte più lente di quelle DES.

Realizzazioni Sw di RSA sono circa 100 volte più lente di quelle DES.

Crittografia omomorfica

Tipo di crittografia per effettuare calcoli matematici su dati crittografati.

Per esempio, avendo due numeri X e Y (cifrati con lo stesso algoritmo omomorfico a partire da due numeri A e B) è possibile calcolare la cifratura della somma di A e B sommando direttamente X e Y, senza bisogno di effettuare la decifratura.

Molto utile per affidare i calcoli su dati interni ad aziende esterne senza problemi di privacy.

Problemi ancora da risolvere:

- Costo computazionale molto oneroso
- Non tutte le operazioni matematiche sono supportate

Generazione e distribuzione delle chiavi

Nei sistemi di crittografia è fondamentale il **problema della gestione** delle chiavi, nelle varie fasi:

- Generazione sicura (richiede una qualche autorità fidata, una certification authority).
- Memorizzazione sicura (es. smart card).
- Reperimento e distribuzione sicura.
- Eventuale revoca della validità di chiavi compromesse.

Generazione e distribuzione delle chiavi

Quante chiavi?

Crittografia a chiave **pubblica**

- una coppia di chiavi per ogni utente. Nel sistema ci sono 2n chiavi (se n è il numero degli utenti).

Crittografia a chiave **segreta**

- ogni persona deve condividere una chiave diversa con ogni altro utente. Nel sistema ci sono n(n-1)/2 chiavi (se n è il numero degli utenti). (n=100, 4950 chiavi)

Come distribuire le chiavi?

- Difficile distribuzione “on line” delle chiavi segrete.
- Le chiavi pubbliche sono di pubblico dominio, possono essere distribuite “on line” da infrastrutture per chiavi pubbliche (PKI).

Nota: per questo motivo sono molto utilizzati metodi a chiave pubblica per scambiarsi chiavi segrete (computazione più leggera) di sessione per la comunicazione tra utenti (es., SSL)

Scambio di oggetti (Massey-Omura)

Problema: Man in the middle

La sicurezza di DES e RSA

DES e RSA sono algoritmi studiati e verificati da molti anni e sono quindi **molto robusti**.

La sicurezza di un sistema di crittografia robusto, che è sopravissuto ad anni di attento scrutinio, dipende (quasi) solo dalla **lunghezza delle chiavi** utilizzate.

Nel caso di un attacco di *forza bruta* (si tenta la decifrazione di un messaggio provando con tutti i possibili valori di chiave):

$$\mathbf{T_{max} = tau * 2n/N}$$

T_{max} = tempo massimo richiesto per scoprire la chiave

tau = tempo richiesto per una verifica

n = lunghezza della chiave (numero di bit)

N = Numero di calcolatori in parallelo

La sicurezza di DES ed RSA

Caso DES:

Lunghezza della chiave in Bits

(valori stimati nel 1995, fonte "Applied Cryptography", B. Schneier)

Cost	**40**	**56**	**64**	**80**	**112**	**128**
$100 K	2 sec.	35 h	1 y	70,000 y	10^{14} y	10^{19} y
$1 M	.2 sec.	3.5 h	37 d	7000 y	10^{13} y	10^{18} y
$10 M	.02sec.	21 m	4 d	700 y	10^{12} y	10^{17} y

Tmax nel caso di DES a diverse lunghezze di chiave con diversi elaboratori di diverso costo.

(si noti che DES è esportato fuori dagli USA a 40 bit)

La sicurezza di DES ed RSA

Caso RSA:

il livello di sicurezza è garantito dal fatto che i numeri da fattorizzare siano molto grandi

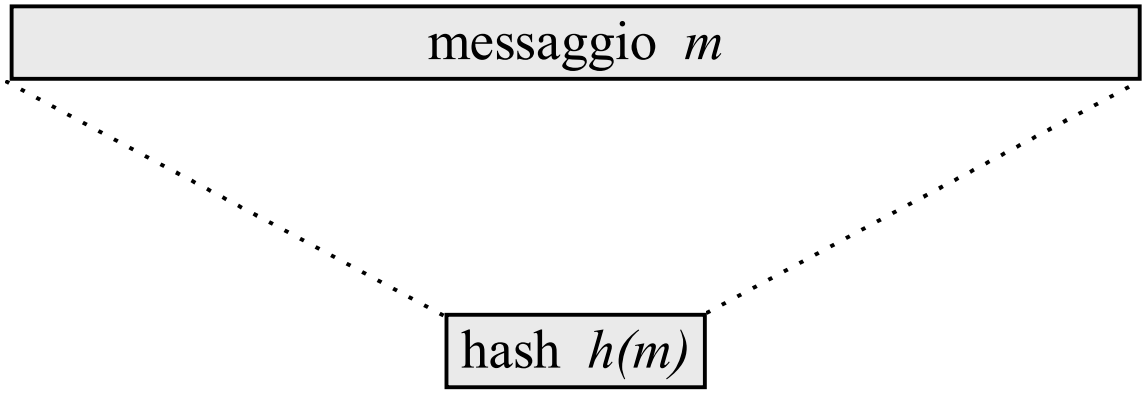
Tempi di fattorizzazione (un Pentium a 100 MHz ha circa 50 MIPS)

# di bits	**Mips-anno per fattorizzare**
512	30,000
768	$2 * 10^{8}$
1024	$3 * 10^{11}$
1280	$1 * 10^{14}$
1536	$3 * 10^{16}$
2048	$3 * 10^{20}$

Funzioni Hash

Le funzioni hash sono molto utilizzate nei sistemi crittografici per migliorare l'efficienza e le prestazioni.

Una funzione hash comprime un input m di lunghezza arbitraria in un output h(m) di lunghezza fissa e di piccole dimensioni.



Caratteristiche funzione hash:

- per ogni messaggio *m* è facile calcolare *h(m)*
- dato *h(m)* è difficile trovare un *m* che lo fornisca
- deve essere difficile trovare due *m* diversi con lo stesso *h(m)*

Impiego degli algoritmi Hash

Password hashing

password cifrata registrata su file

hash della password cifrata su file

Integrità dei messaggi

hash usata per generare MIC del messaggio

hash (messaggio+password)

Impronta dei file

MIC di un file per accertarsi della sua integrità

Efficienza delle firme digitali

si firma solo l'hash del documento che si vuole inviare.

Le funzioni Hash non sono sufficienti

Problema

Online sono presenti interi dizionari di database con una moltitudine di hash.

Hash h(m)

Esempio:

https://md5decrypt.net/

Una soluzione

Una delle possibili soluzioni utilizzate è effettuare il salting dell'hash.

hash h(m + valore casuale)

Encrypt-then-MAC or MAC-then-encrypt?

Nel caso volessimo sia cifrare che autenticare un messaggio, è indispensabile fare attenzione all'ordine in cui queste operazioni vengono eseguite

Infatti, la soluzione MAC-then-encrypt è suscettibile a chosen ciphertext attack:

https://moxie.org/blog/the-cryptographic-doom-principle/

Bisogna ***SEMPRE* usare la modalità Encrypt-then-MAC!**

La crittografia (come la sicurezza più in generale) è una disciplina in cui piccoli dettagli implementativi possono fare una differenza enorme...

Sistemi di crittografia a *chiave pubblica*
Distribuzione delle chiavi

L'elenco delle chiavi ***pubbliche*** è un potenziale punto debole per la sicurezza del sistema

Esiste il problema della **autenticità delle chiavi pubbliche**:

> un utente può in malafede pubblicare una chiave a nome di un altro e utilizzarla per sostituirsi a lui

Si ricorre a terze parti, dette **Certification Authority**, che garantiscono l'integrità e l'autenticità dell'elenco delle chiavi pubbliche

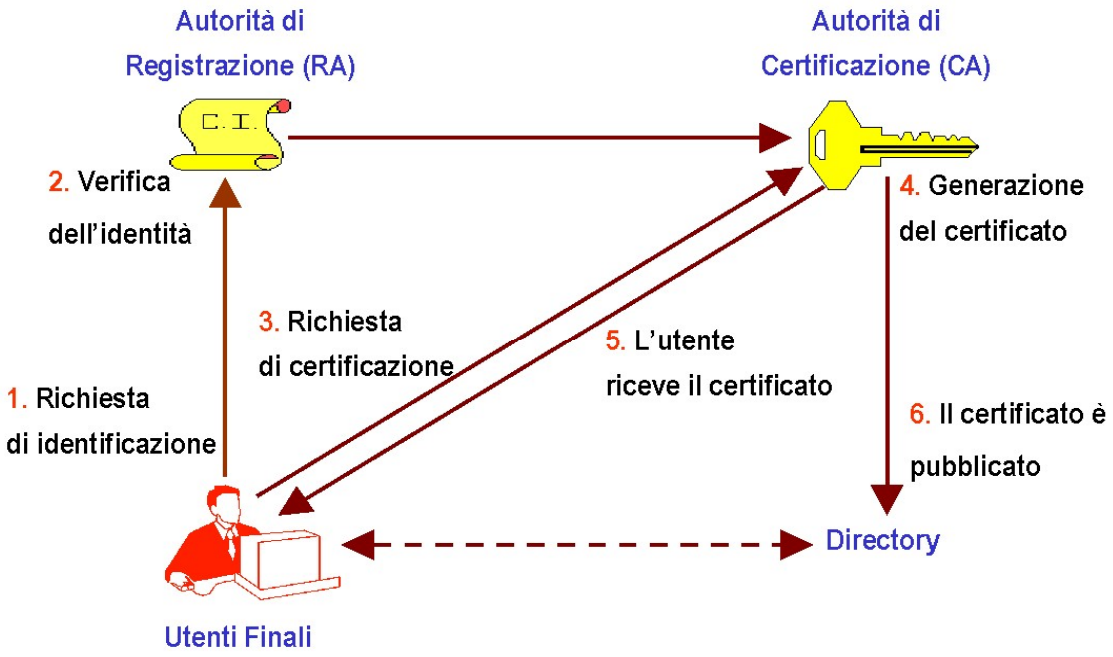
> la Certification Authority deve essere al di sopra di ogni sospetto, compatibilmente con il livello di sicurezza desiderato

Tali Certification Authority (CA) costituiscono componenti essenziali delle infrastrutture a chiave pubblica (Public Key Infrastructure - PKI)

Infrastrutture a chiave pubblica (PKI)

Funzionalità:

- generazione e distribuzione sicura delle chiavi (uso di certificati)
- validazione di chiavi (e certificati)
- aggiornamento e revoca di chiavi e certificati

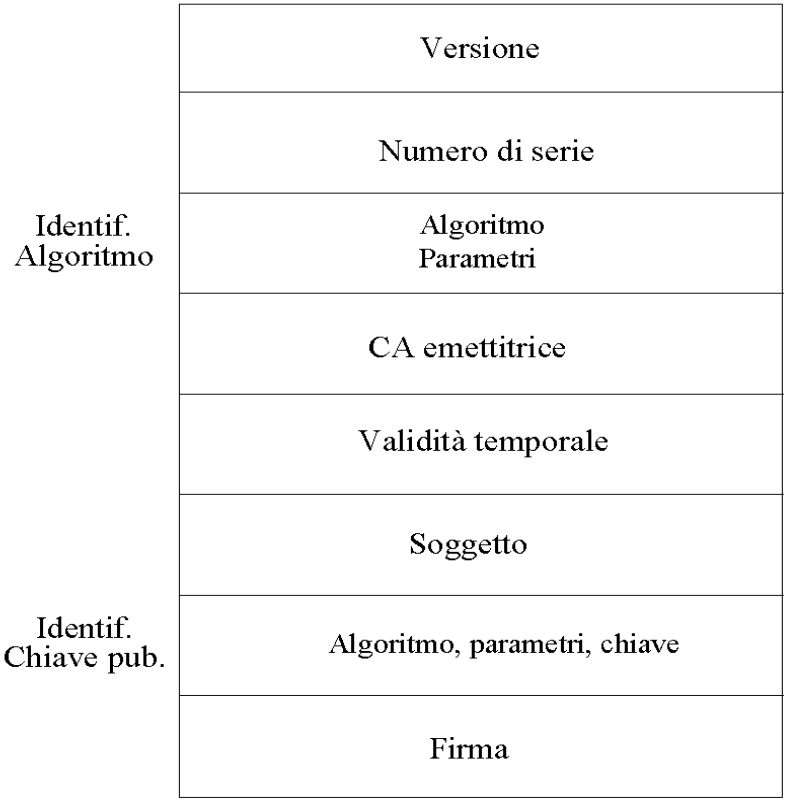


CA associa chiavi pubbliche a identità fisiche attraverso la creazione di un certificato elettronico. Distribuzione, aggiornamento e **revoca** di certificati.

Certificato elettronico: struttura

Un certificato composto da molti campi, tra cui:

- Versione distingue vari formati
- Numero di serie identifica in modo univoco il certificato
- CA è la Certification Authority che ha prodotto il certificato
- Il certificato ha validità limitata nel tempo (data inizio e fine)
- Soggetto è nome utente a cui si riferisce il certificato
- La chiave è quella pubblica dell'utente
- Firma è la firma di tutto il certificato da parte della CA



Formato standard: X.509

Certificato elettronico: uso

Il certificato si compone di molte parti, ma le più importanti sono:

- nome dell'utente
- chiave pubblica dell'utente

La CA firma (con la sua chiave privata) il certificato (ne firma un hash).

Chiunque può usare la chiave pubblica della CA per verificare l'autenticità del certificato (e quindi verificare l'integrità della chiave pubblica dell'utente).

E se si deve annullare un certificato (per esempio per la perdita della propria chiave privata, o per la cessazione improvvisa di una posizione in una organizzazione): problema della revoca.

Modelli di Revoca

Pull. L'utente finale richiede lo stato di revoca del certificato

Push. Lo stato di revoca di un certificato viene comunicato agli interessati da parte della CA o di un servizio delegato da essa

Esempi di Meccanismi:

- liste di revoca (Certificate Revocation List – **CRL**).
  La CA a tempi predefiniti pubblica sulla directory una lista firmata contenenti tutti i certificati revocati. Un certificato revocato viene rimosso dalla lista solo allo scadere della sua validità temporale.
- online certificate status protocol (**OCSP**).
  Lo stato di revoca di uno specifico certificato viene richiesto *on demand* a un servizio on line.

Problemi: tradeoff tra requisiti di sicurezza delle applicazioni e carico computazionale (per l'utente finale, per la CA, per la directory)

Esempio: carico computazionale per l'utente finale

Modelli di fiducia

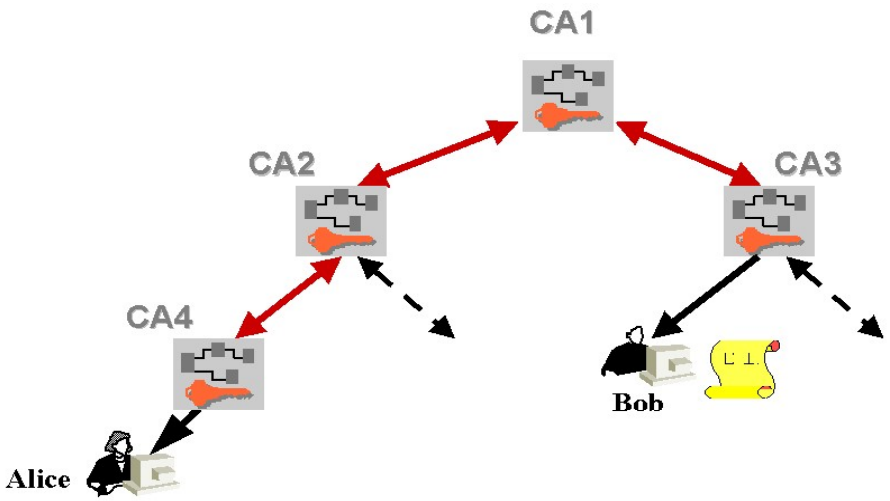
Problema:

Alice deve verificare l'autenticità di un messaggio inviato da Bob. Deve effettuare:

- recupero del certificato di Bob
- convalida del certificato in termini di autenticità e stato

necessità di un "cammino di certificazione" tra Alice e Bob => problemi di efficienza nel ritrovamento e convalida di un cammino di certificazione

Modelli di fiducia



Cammino di Certificazione:

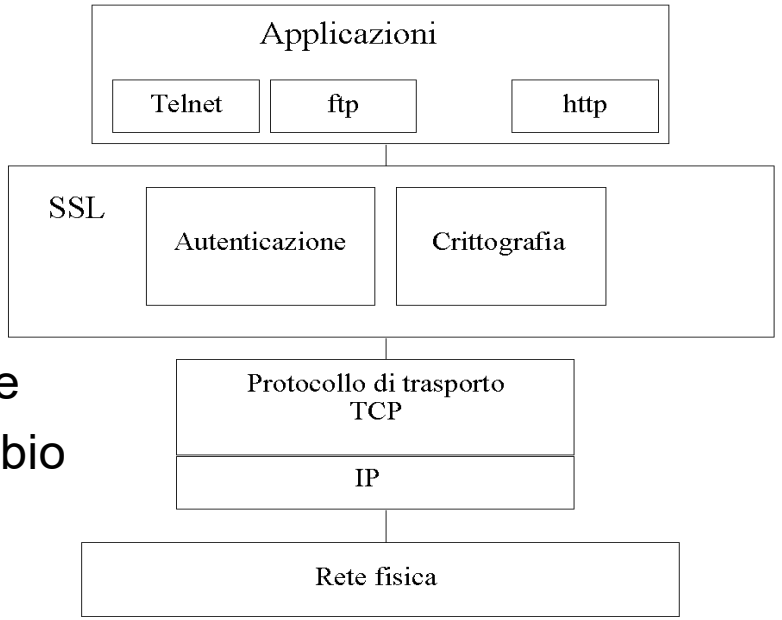
Subject: Alice	Subject: CA 4	Subject: CA 2	Subject: CA 3	Subject: Bob
CA 3 Public Key	CA 3 Public Key	CA 3 Public Key	CA 3 Public Key	Bob Public Key
Issuer: **CA 4**	Issuer: **CA 2**	Issuer: **CA 1**	Issuer: **CA 1**	Issuer: **CA 3**

I ritrovamento e la convalida di un cammino di certificazione dipendono dal tipo di modello di fiducia instaurato dalle diverse CA

Transport Layer Security (TLS)

Inizialmente proposto da Netscape col nome di Secure Socket Layer (SSL), e successivamente standardizzato in ambito IETF. Utilizza una combinazione di soluzioni per le varie funzioni da svolgere.

Nella suite OSI si colloca logicamente nello strato di Sessione. *Non richiede necessariamente una modifica delle applicazioni.*



TLS crea una sessione, cioè un'associazione tra il Client e il Server, che permette lo scambio dei dati tra C/S in modo sicuro (supporto ad autenticazione, privacy e integrità).

Lo scambio messaggi può anche avvenire su più connessioni all'interno della stessa sessione.

Uso di TLS per traffico sicuro su Web (HTTPS), per VoIP (SIP), per fare VPN.

TLS: schema di funzionamento

TLS prevede 3 fasi principali:

1. Negoziazione fra le parti per scegliere algoritmi da usare (si negoziano i 3 algoritmi di *key exchange*, di *cifratura pubblica* e di *autenticazione tramite hash*). Server sceglie tra le proposte del Client (non è detto che il Server scelga sempre l'algoritmo più sicuro, quindi il Client non deve mai proporre algoritmi che ritiene non sufficientemente sicuri)
2. Scambio delle chiavi e (mutua) autenticazione. Il Server invia il proprio certificato al Client, con cui si può condividere in modo sicuro una chiave simmetrica di sessione.
3. Scambio dei messaggi usando cifratura simmetrica (tramite la chiave di sessione sopra concordata)

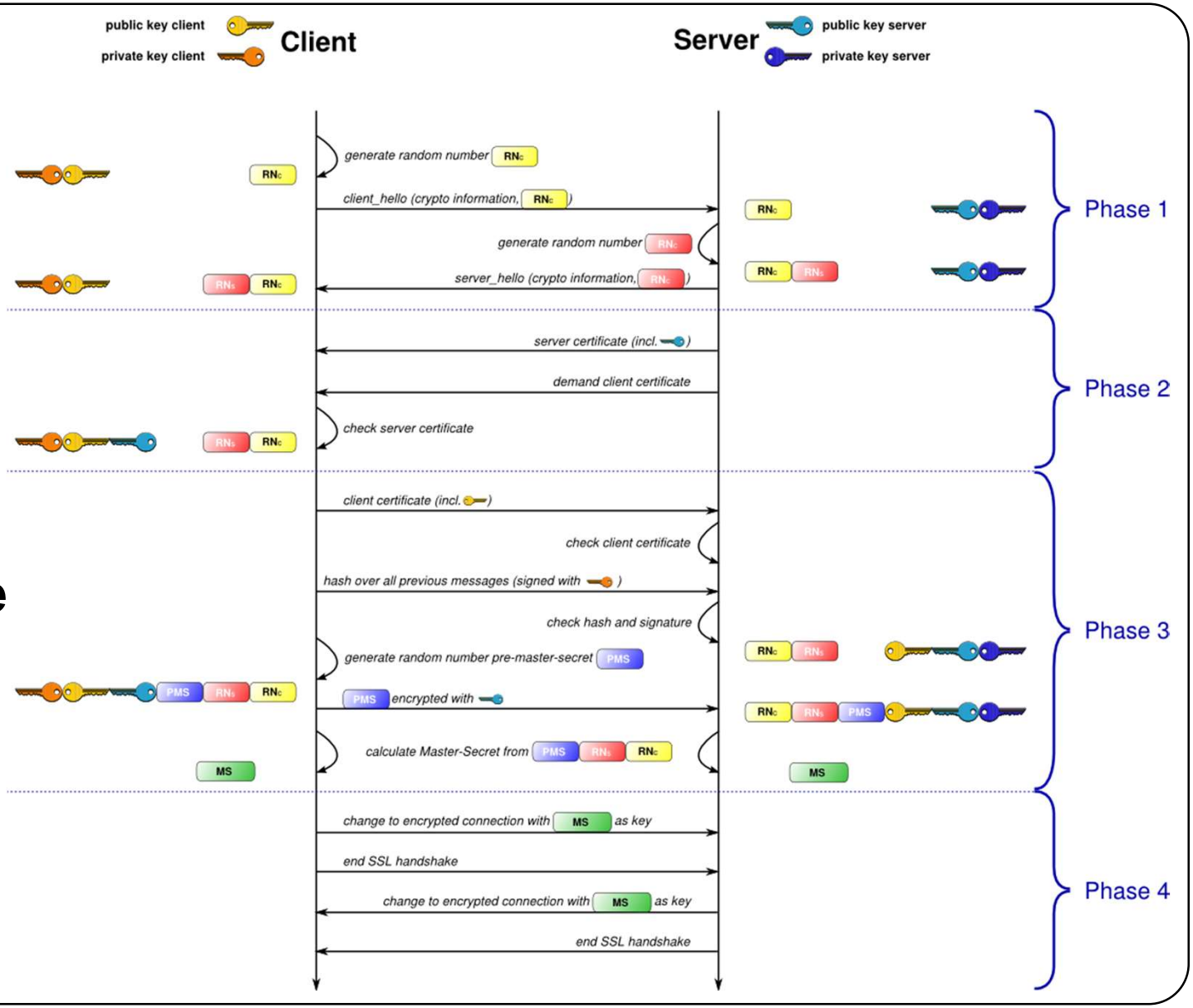
TLS: schema di funzionamento

TLS si basa su alcuni protocolli crittografici, i più importanti sono:

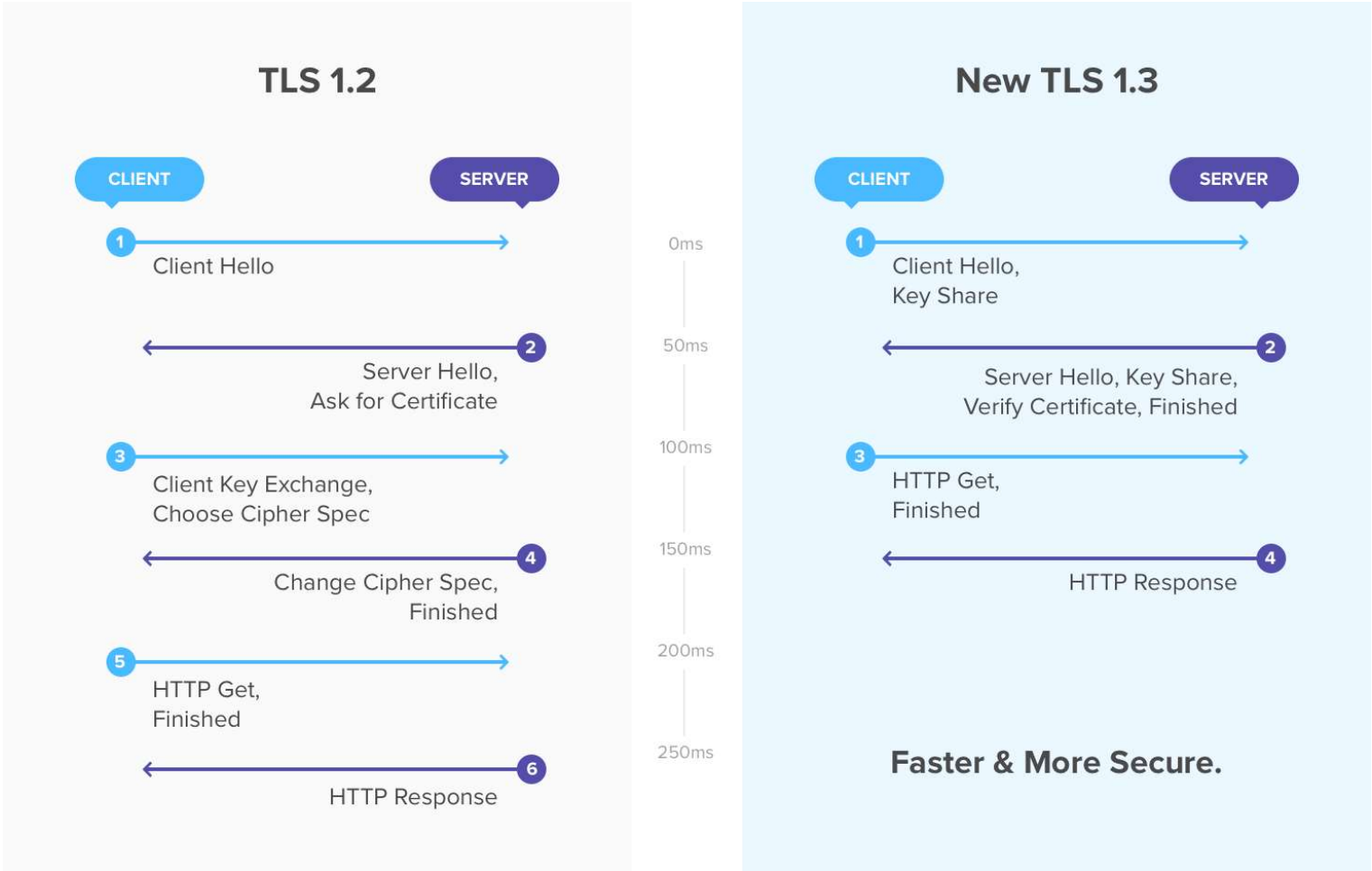
- TLS Handshake Protocol (THP)
  - negozia gli algoritmi pubblici da utilizzare per lo scambio delle chiavi (es., RSA, Diffie Hellman)
  - si occupa dell'autenticazione di server e client (opzionale) (mediante certificati X.509)
  - definisce la chiave segreta-simmetrica (master key) utilizzata da Client e Server per lo scambio dei dati nella fase successiva (si creano diverse chiavi per diverse funzioni crittografiche, MAC e cifratura)

- TLS Record Protocol (TRP)
  - Frammentazione messaggio in blocchi (con eventuale padding)
  - Aggiunta codice MAC per ogni blocco (con chiave sopra negoziata)
  - Cifratura/decifrazione dei blocchi (con chiave sopra negoziata)

TLS Handshake



Handshake semplificato in TLS 1.3



Si veda anche https://tls13.ulfheim.net/

Esempio di codice client in OpenSSL 1/6

```
#include <openssl/crypto.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#define CA_CERT_LOCATION «/var/lib/ca_certificates»


…
  SSL_CTX *ctx; SSL *ssl; X509 *server_cert;

  SSL_library_init(); /* Inizializzazione libreria */
  SSL_load_error_strings(); /* Caricamento stringhe di errore */

  /* Creazione struttura SSL_CTX */
  if ((ctx = SSL_CTX_new(TLSv1_method())) == NULL) {
    ERR_print_errors_fp(stderr);
    exit(1);
  }
```

Esempio di codice client in OpenSSL 2/6

```
/* eventuale verifica certificati client (opzionale) */
…

/* Caricamento certificati Trusted CA, per verificare il certificato del Server */
if(!SSL_CTX_load_verify_locations(ctx, CA_CERT_LOCATION, NULL)) {
  ERR_print_errors_fp(stderr);
  exit(EXIT_FAILURE);
}

/* Per controllare il certificato in fase di handshake (se i controlli non vanno a buon fine la
connessione non viene stabilita) */
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
SSL_CTX_set_verify_depth(ctx, 1); /* Massima lunghezza catena di certificazione */

/* Setup socket attiva sd */
…
```

Esempio di codice client in OpenSSL 3/6

```
/* Creazione struttura SSL */
if ((ssl = SSL_new(ctx)) == NULL) {
  ERR_print_errors_fp(stderr); exit(EXIT_FAILURE);
}

/* Assegnazione della socket attiva sd alla struttura ssl */
if ((err = SSL_set_fd(ssl, sd)) == 0) {
  fprintf(stderr,"errore SSL_set_fd"); exit(EXIT_FAILURE);
}

/* TLS client handshake */
if ((err = SSL_connect(ssl)) == -1) {
  int errorcode = SSL_get_error(ssl, err);
  switch(errorcode){
    …
  }
}
```

Esempio di codice client in OpenSSL 4/6

```
/* Ottengo certificato server */
if ((server_cert = SSL_get_peer_certificate(ssl)) != NULL) {
  result = SSL_get_verify_result(ssl); /* Verifico il certificato del Server */
  switch(result) {
    case X509_V_OK: puts("Certificato OK"); break;
    case X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT:
      puts("Certificato AUTO FIRMATO"); /* COSA FARE? */
    case X509_V_ERR_CERT_UNTRUSTED:
      puts("Certificato firmato da CA NON RICONOSCIUTA!"); exit(EXIT_FAILURE);
    case X509_V_ERR_CERT_REVOKED:
      puts("Certificato REVOCATO!"); exit(EXIT_FAILURE);
  }
  X509_free(server_cert); /* Dealloco la memoria */
} else {
  puts("Il Server non possiede un certificato"); exit(EXIT_FAILURE);
}
```

Esempio di codice client in OpenSSL 5/6

```
/* Invio dati al Server */
if ((err = SSL_write(ssl, request_str, strlen(request_str))) == -1){
  ERR_print_errors_fp(stderr); exit(EXIT_FAILURE);
}

/* Ricezione dati dal Server */
memset(buff, 0, N);
if ((err = SSL_read(ssl, buff, N)) == -1){
  SSL_shutdown(ssl);
  close(sd);
  ERR_print_errors_fp(stderr);
  exit(EXIT FAILURE);
}

…
```

Esempio di codice client in OpenSSL 6/6

```
  /* Eseguo shutdown connessione TLS */
  if ((err = SSL_shutdown(ssl)) == -1){
    ERR_print_errors_fp(stderr); exit(EXIT_FAILURE);
  }

  /* Chiudo socket attiva */
  close(sd);

  /* Libero strutture SSL e SSL_CTX */
  SSL_free(ssl);
  SSL_CTX_free(ctx);

  return 0;
}
```

Let's Encrypt - Free SSL/TLS Certificates

Let’s Encrypt è una Certification Authority (CA) no-profit sviluppata dall’Internet Security Research Group (ISRG).

Rilascia gratuitamente certificati Domain Validated (DV) con una durata di 90 giorni attraverso un processo automatizzato.

Let's è stata annunciata pubblicamente nel Novembre del 2014.

Il protocollo utilizzato per l'autenticazione e il rilascio dei certificati si chiama Automated Certificate Management Environment (ACME), è un protocollo del tipo challenge-response.

Data	Certificati rilasciati	Data	Certificati rilasciati
27 Novembre 2016	20 milioni	6 Agosto 2018	100 milioni
12 Dicembre 2016	24 milioni	14 Settembre 2019	380 milioni
28 Giugno 2017	100 milioni	24 Ottobre 2019	837 milioni

Not so easy, tiger!

Attenzione! La realizzazione di applicazioni sicure basate su SSL è ***MOLTO*** più difficile di quanto vi possa sembrare. Si vedano:

Mozilla Server Side TLS Guideline

https://wiki.mozilla.org/Security/Server_Side_TLS

Georgiev et al., “The most dangerous code in the world: validating SSL certificates in non-browser software”

http://crypto.stanford.edu/~dabo/pubs/abstracts/ssl-client-bugs.html

https://www.openssl.org/news/vulnerabilities.html

https://xkcd.com/1354/

http://heartbleed.com/

https://blog.cryptographyengineering.com/2014/10/15/attack-of-week-poodle/

https://www.isotoma.com/blog/2008/05/14/debians-openssl-disaster/

Intranet e Internet

Internet rete intrinsecamente insicura

Collegamento di Intranet aziendali a Internet è un problema molto sentito, soprattutto per:

- economicità (basso costo collegamenti)
- mercato globale (WWW)
- supporto mobilità utenti

Principio fondamentale

se si vuole essere totalmente sicuri, è meglio non essere connessi

Fattori di perdita di sicurezza in Internet

TCP/IP come sistema aperto

uso di risorse esterne per routing

vulnerabilità intrinseche dei servizi e protocolli

estrema complessità di meccanismi di controllo

Facilità di monitoraggio dell'attività di rete:

comunicazioni in chiaro

Controllo degli accessi e autenticazione utenti

spesso basati su password (statiche e riusabili)

Connessioni di rete tramite risorse esterne

linee condivise e router di terzi

Intranet e Internet

Possibilità di accedere ai servizi di rete:

- *accesso dall'interno verso l'esterno*
- ***accesso dall'esterno verso l'interno,* senza compromettere il sistema interno**

Per le **organizzazioni commerciali** o **bancarie** diventa vitale trovare soluzioni accettabili.

Politiche e meccanismi di ***separazione, per collegare in modo sicuro*** gli **ambienti interni (Intranet)** *e* ***Internet*** *(uso di sistemi firewall e VPN)*

Firewall

Un **firewall** garantisce la sicurezza del collegamento di una Intranet verso Internet. (Non consideriamo i personal firewall sulle macchine Windows)

Un firewall è un sistema costituito da molti componenti che:

- rappresenta **l'unico punto di contatto** della rete **interna** con l'**esterna**
- filtra e **controlla** tutto il traffico tra le due reti
- concentra i **meccanismi di sicurezza**
- impone la **politica di sicurezza** della organizzazione
- nasconde informazioni della rete interna
- registra eventi (**logging**) ed elabora statistiche sul traffico di rete (**auditing**)

Si noti l'importante separazione tra politiche e meccanismi.

Attenzione alla scelta dei servizi che devono transitare attraverso il firewall.

Firewall: politiche

Il **firewall** deve implementare una **politica di accesso** in modo **separato** e **concentrato:**

- ☹ senza firewall le stesse funzioni vengono ottenute attraverso la cooperazione di tutti gli host

Firewall: problemi

Un firewall non risolve tutti i problemi:

- ☹ un **firewall** restringe la possibilità di accesso a servizi
- ☹ la topologia di rete può essere inadeguata a un firewall
- ☹ non protegge contro i **virus**
- ☹ problemi di **attacchi interni** (*compromesso tra sicurezza e funzionalità)*
- ☹ attenzione alle vie di accesso secondarie (**backdoor**) (es. modem)
- ☹ come punto **concentrato** di affidabilità, ma può diventare un **collo di bottiglia** (bottleneck)

Firewall: Aspetti progettuali

Alcuni aspetti da tenere in considerazione nel progetto di un firewall:

- autenticazione (di utenti e servizi)
- autorizzazione (nell'accesso a servizi e risorse)
- esigenze da soddisfare (quali servizi abilitare)
- architettura del sistema

Firewall: Autenticazione e Autorizzazione

Metodologie di autenticazione

Tecniche di autenticazioni robuste, anche basate su diversi meccanismi:

password (anche usa e getta - one-time password)

smart card

dati biometrici (impronta digitale, venatura retina, etc.)

Due politiche di autorizzazione opposte:

tutto ciò che non è espressamente permesso è vietato

- maggiore sicurezza, ma più difficile da gestire

tutto ciò che non è espressamente vietato è permesso

- minor sicurezza, ma più facile da gestire

Firewall: problemi di efficienza

Il firewall comporta un'inefficienza nei servizi che sono disponibili, con un ritardo nei tempi di risposta.

Il firewall potrebbe *anche diventare il collo di bottiglia dell'intero sistema*

Considerazioni generali

- grossi oggetti sono difficili da gestire ('*grande non è bello*')
- usare risorse dedicate solo ai meccanismi di sicurezza

Tipi di Firewall

Ci sono 3 tipi principali di firewall:

Packet filtering firewall

Stateful inspection firewall

Application proxy firewall (o application-level gateway)

Packet filtering firewall

Il **packet filtering firewall** esegue un semplice filtraggio a livello di rete. Il traffico filtrato sulla base dei campi contenuti nel datagramma IP:

sourceIP, sourcePORT, destinationIP, destinationPORT

Si possono così *escludere alcuni host come mittenti o come destinatari ed escludere alcuni servizi*

tipo	Indirizzo destination	Indirizzo source	Porta destin.	Porta source	Azione
TCP	137.204.57.33	*	23	>1023	permit
TCP	137.204.57.32	*	25	>1023	permit
TCP	137.204.57.34	*	25	>1023	permit
TCP	137.204.57.31	137.2.5.30	119	>1023	permit
TCP	*	*	*	*	deny

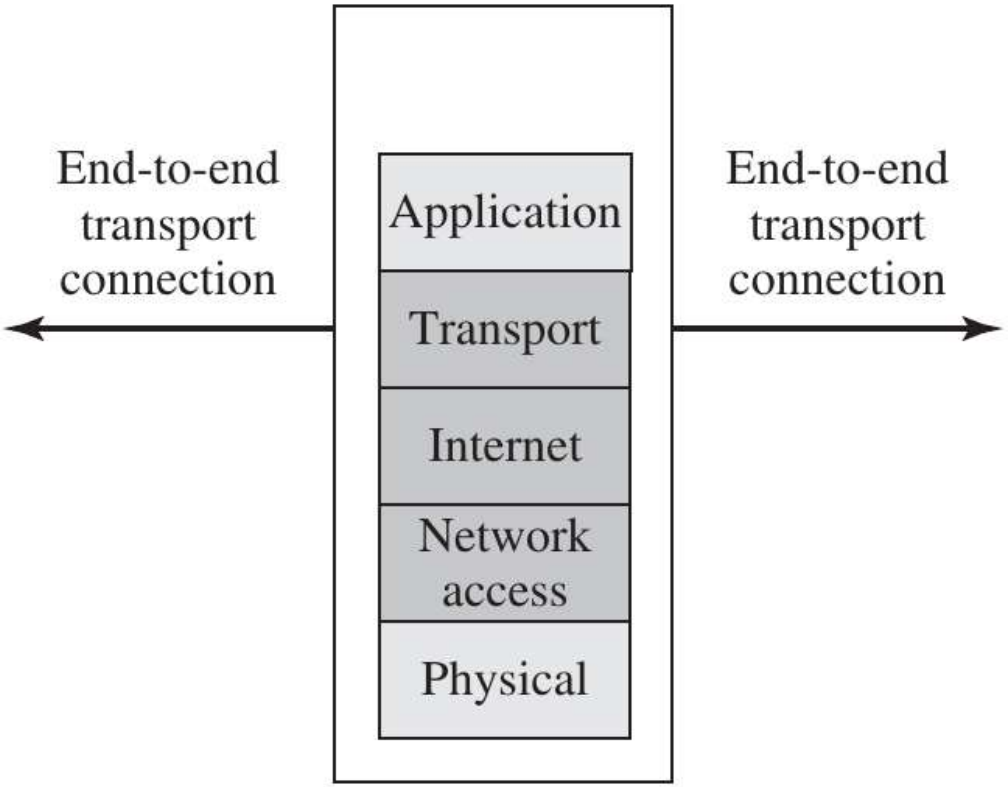
Packet filtering firewall è **molto efficiente** ma limitato, con alcuni **problemi**:

- ☹ difficoltà in caso di servizi RPC con più interfacce
- ☹ difficoltà di specificare regole in modo compatto
- ☹ mancanza di logging/auditing
- ☹ nessuna protezione a livello applicativo

Packet filtering firewall

Il **packet filtering firewall** analizza pacchetto per pacchetto, filtrando il traffico a seconda di semplici regole

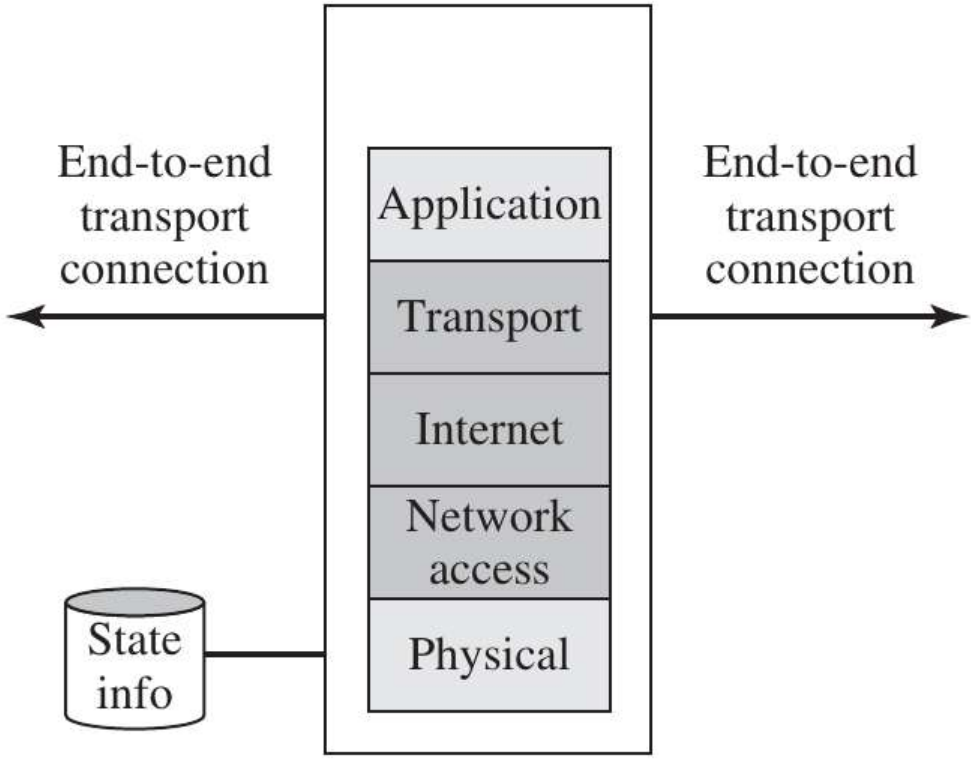
Nessuna informazione sullo stato delle comunicazioni è disponibile per il filtraggio del traffico



Stateful inspection firewall

Lo **stateful inspection firewall** tiene traccia delle connessioni TCP aperte

Controllando che i pacchetti in arrivo dalla rete esterna corrispondano allo stato della corrispondente connessione TCP (es. numeri di sequenza, ecc.) è in grado di fornire protezione aggiuntiva rispetto al packet filtering firewall



Application proxy firewall

I problemi del packet filtering si possono superare con dei proxy, *cioé gestori ad-hoc per consentire il trattamento solo di uno specifico servizio*

Un **proxy** server è un'*applicazione* col compito di mediare il traffico tra rete esterna e interna per consentire accesso a un servizio specifico.

vantaggi

- ☺ filtra *servizi* e *protocolli* (proxy per servizi, esempio: telnet, ftp, e-mail,…)
- ☺ supporta *autenticazione* robusta e logging/auditing
- ☺ semplifica le regole del filtering
- ☺ garantisce riservatezza alla rete interna
- ☺ incide positivamente sul costo. I proxy devono essere concentrati sul solo firewall e non distribuiti su tutti gli host della rete

svantaggi

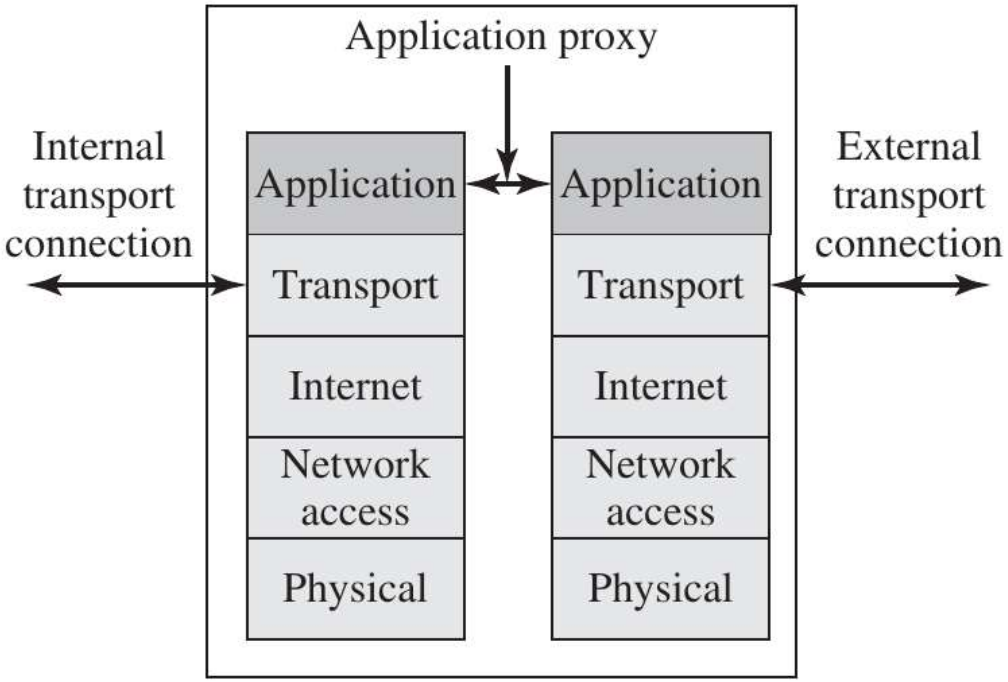
- ☹ connessioni con host interno a due passi, quindi **perdita di trasparenza a causa del firewall,** *a meno di modificare i client*

Application proxy firewall

L'**application proxy firewall** (anche noto come **application-level gateway**) lavora a livello applicativo e non di rete e/o trasporto

Esso verifica che applicazioni e utenti che cercano di comunicare con l'esterno abbiano effettivamente i permessi necessari

Le applicazioni non possono comunicare direttamente con l'esterno ma devono passare per il proxy



Firewall: posizionamento

Si possono installare i firewall in molti punti della rete, a seconda delle esigenze:

Personal firewall, per proteggere le comunicazioni di un singolo utente,sui computer degli utenti

Host-based firewall, per proteggere le comunicazioni di una singola macchina, su ciascun server

Bastion host macchina sicura dedicata al controllo del sistema e del traffico (per es. **auditing** ossia verifica e traccia degli eventi nel sistema), che rappresenta il punto più sicuro della rete e tipicamente ospita strumenti sofisticati come application-level gateway

Architetture di Firewall

Diversi tipi di architetture:

Screening router firewall di tipo packet filtering (stateless o stateful) che esegue sul router che divide la rete interna da quella esterna – soluzione tipica in ambienti SOHO

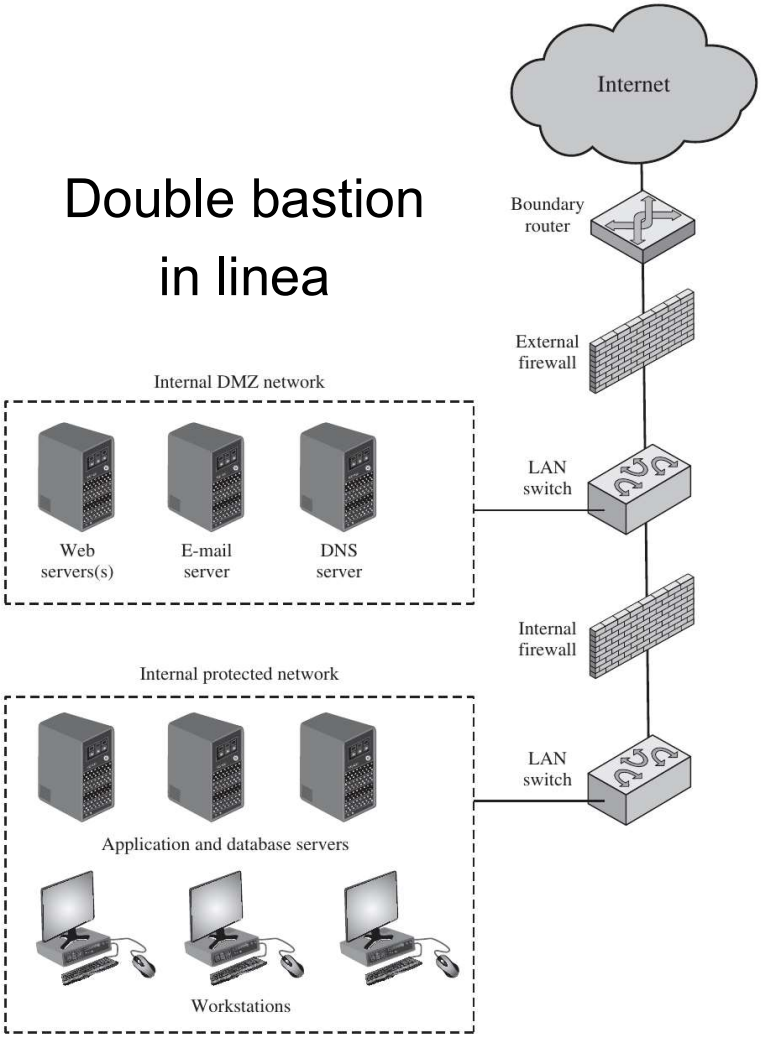
Single bastion bastione che divide la rete interna da quella esterna – soluzione tipica in aziende medio-piccole

Double bastion i server sono posti in una zona «smilitarizzata» (DMZ) posta tra la rete esterna e quella interna; ogni zona è separata dalle altre da un bastione – soluzione tipica in aziende grandi

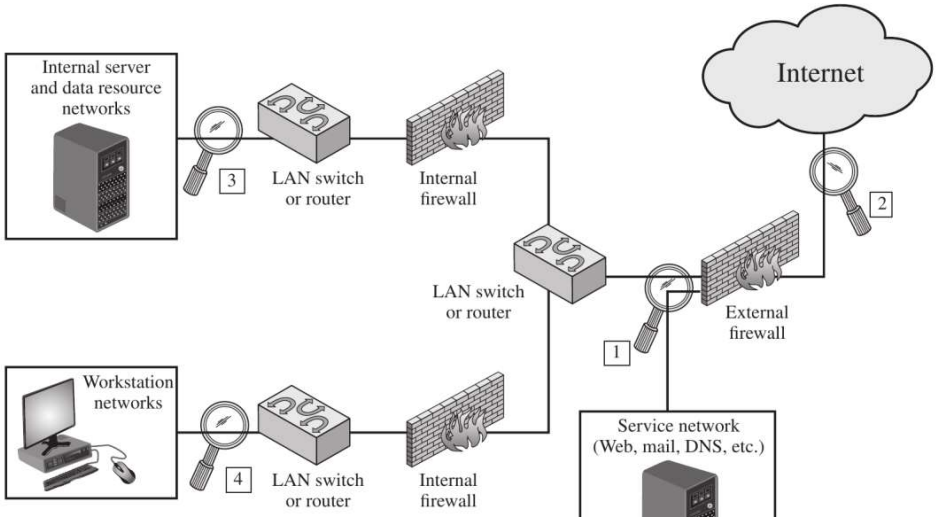
Distributed firewall due zone «smilitarizzate» e protezioni a vari livelli – soluzione usata in alcune aziende grandi

Double bastion

Double bastion in linea



Applicazioni e servizi accessibili sia dalla rete interna che da quella esterna sono installati in un'apposita parte della rete detta «zona smilitarizzata» (DMZ).



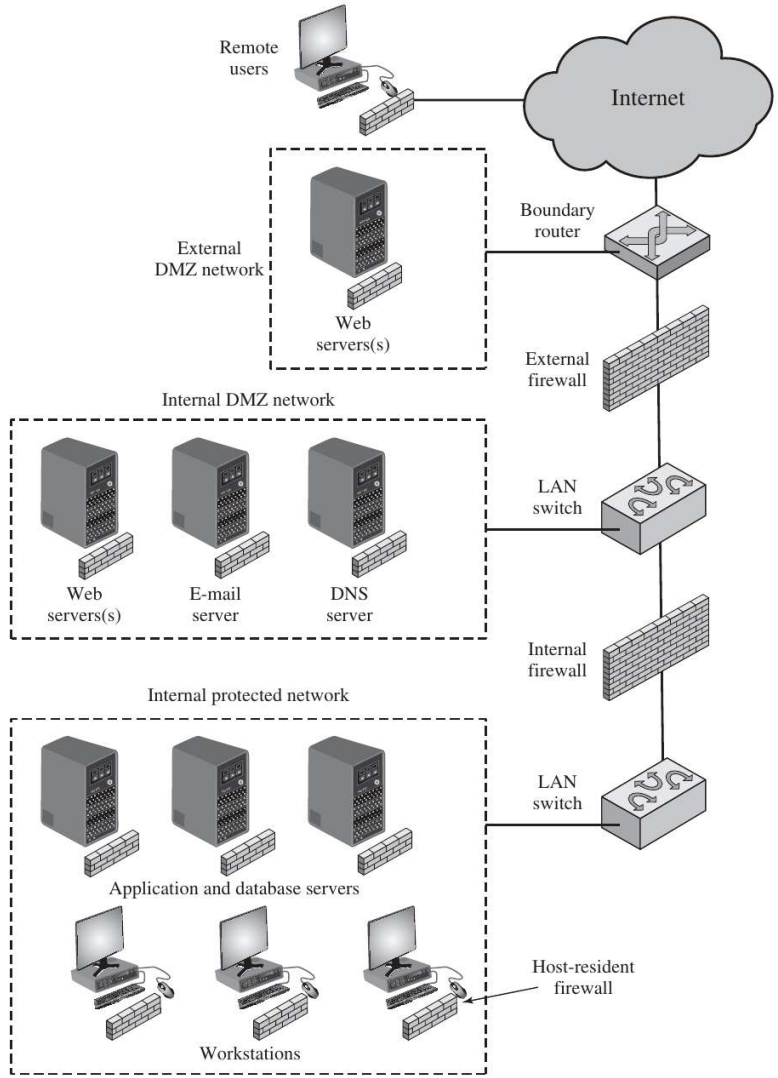
Double bastion a T

Distributed firewall

Due diverse «zone smilitarizzate»:

- External DMZ per applicazioni e servizi accessibili dall'esterno
- Internal DMZ per applicazioni e servizi accessibili dalla rete interna (Intranet)

Di solito questa soluzione è resa gestibile da strumenti di configurazione sofisticati che automatizzano la gestione dei vari firewall installati nella rete



Honeypot

Un honeypot è un sistema (hardware/software) utilizzato come "trappola" o "esca" a fini di protezione contro gli attacchi informatici.

Solitamente consiste in un computer o un sito che sembra essere parte della rete e contenere informazioni preziose, ma che in realtà è ben isolato e non ha contenuti sensibili o critici; potrebbe anche essere un file, un record, o un indirizzo IP non utilizzato.

Classificazione sul contesto di utilizzo: di produzione, di ricerca.

Classificazione basata sulla modalità d'iterazione: puri, ad alta interazione, a bassa interazione.

Anche siti web o chatroom possono essere considerati honeypot.

Intrusion Detection Prevention System (IDPS)

Intrusion Detection System (IDS)

Monitora continuamente una risorsa al fine di rilevare attività dannose in corso e reagire ad esse.

Esistono diverse classi di IDS:

- Host based
- Network based sistemi di sicurezza reattiva.

Intrusion Prevention System (IPS)

La prevenzione delle intrusioni inizia con il loro rilevamento. Quando un IPS rileva un attacco, mira ad impedire a un utente malintenzionato di accedere a risorse critiche preventivamente.

Ad esempio: può rifiutare pacchetti di dati, impartire comandi a un firewall e persino interrompere connessioni.

VPN (Virtual Private Network)

Una VPN realizza una Intranet privata virtuale al di sopra della rete Internet. Macchine di sottoreti diverse all'interno di una stessa organizzazione possono cooperare direttamente.

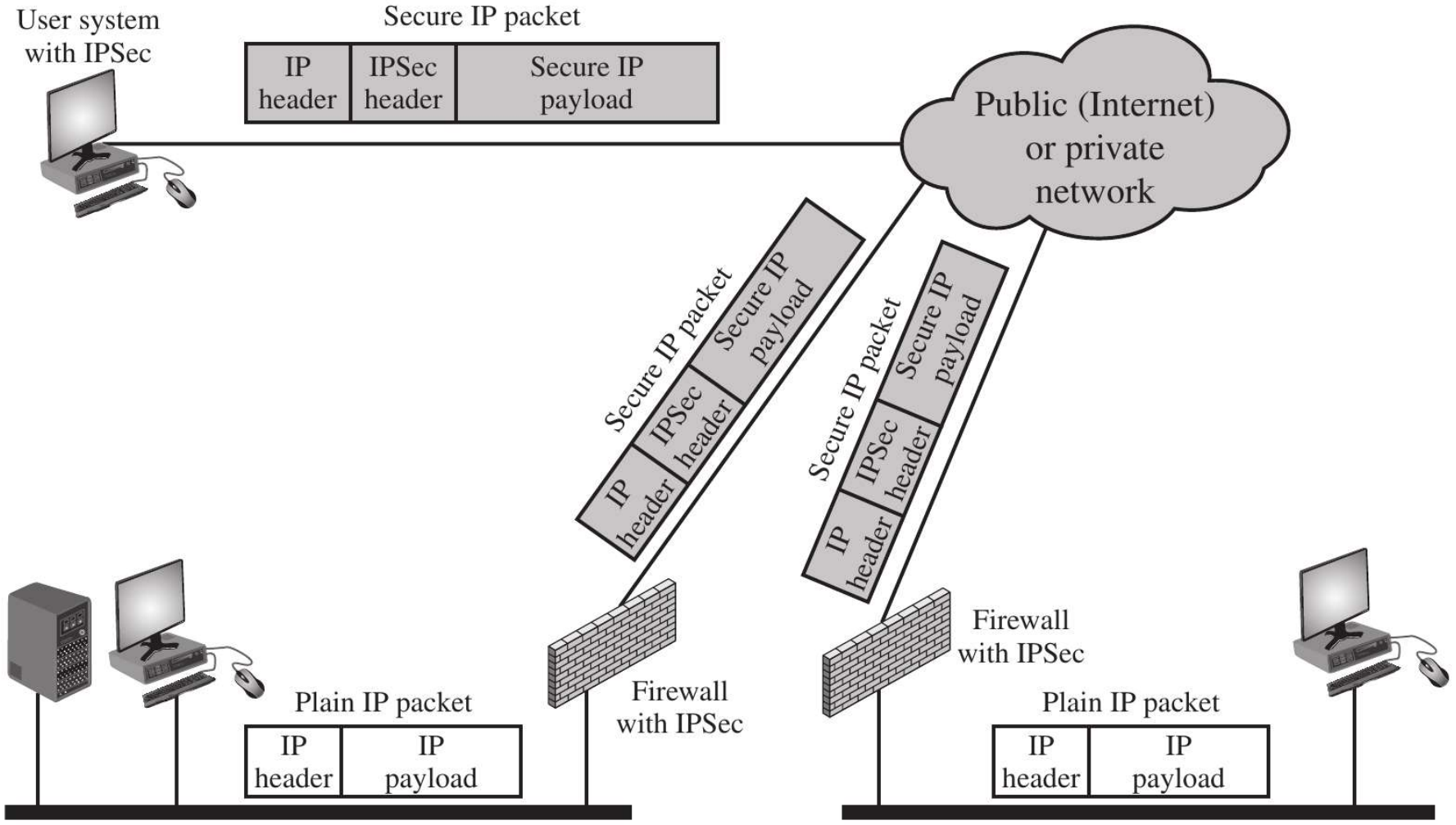
Vantaggi VPN:

- Trasparenza per utenti
- Supporto alla mobilità utenti
- Economicità del collegamento Internet

Tecnologie. I protocolli più diffusi per creare VPN sono:

IPsec, SSL (TLS), PPTP (Point to Point Tunneling Protocol) di Microsoft.

VPN (Virtual Private Network)



Applicazioni

Crittografia, firewall, VPN, ecc. sono solo dei meccanismi. Per realizzare applicazioni e sistemi sicuri è necessario considerare gli aspetti di sicurezza *a livello di progetto e implementazione*:

Patching, patching, patching

- Attenzione a politiche di aggiornamento di sistema operativo, librerie e runtime (es. JVM)

https://blog.cryptographyengineering.com/2017/09/15/patching-is-hard-so-what/

Limitare i permessi delle applicazioni:

- Uso dei minimi privilegi (es. POSIX ACL and POSIX capabilities in Linux)
- Isolamento applicazioni (es. namespaces e chroot/container in Linux)

https://jvns.ca/blog/2016/10/10/what-even-is-a-container/

- Sandboxing (es. seccomp in Linux, pledge in OpenBSD)
- Limite massima disponibilità di risorse (es. cgroups in Linux)
- Possibile uso di soluzioni MAC (es. SELinux)
- Gestione di microservizi e di reti virtuali (es. Istio e Cilium)

Applicazioni

Adozione implementazione sicure di protocolli crittografici:

- Adozione protocolli e librerie best practice
- Uso di CSPRNG robusti

https://sockpuppet.org/blog/2014/02/25/safely-generate-random-numbers/

- Attenzione a timing e side-channel attack

Validazione input:

- Buffer overflow (https://avicoder.me/2016/02/01/smashsatck-revived/ http://arstechnica.com/security/2015/08/how-security-flaws-work-the-buffer-overflow/)
- Cross-side scripting (XSS)
- SQL injection (https://xkcd.com/327/)
- Session hijacking (http://railscasts.com/episodes/356-dangers-of-session-hijacking)

Ricordate sempre che «security as an afterthought is a proven recipe for disaster»!!!

Riferimenti su Internet Security

- Codici e Segreti, S. Singh, Rizzoli.
- **Cryptography Engineering: Design Principles and Practical Applications**, N. Ferguson, B. Schneier, T. Kohno, Wiley (ma leggetevi anche: https://sockpuppet.org/blog/2013/07/22/applied-practical-cryptography/)
- Understanding Cryptography: A Textbook for Students and Practitioners, C. Paar, J. Pelzl, B. Preneel, Springer.
- Computer Security: Principles and Practice (3rd Ed.), W. Stallings, Prentice Hall
- Cryptography and Network Security: Principles and Practice (6th Ed.), W. Stallings, Prentice Hall.
- Network Security Essentials Applications and Standards (5th Ed.), W. Stallings, Prentice Hall.
- Network Security: Private Communication in a Public World (2nd Ed.), C. Kaufman, R. Perlman, M. Speciner, Prentice Hall.
- Security in Computing (4th Ed.), C. Pfleeger, S. Pfleeger, Prentice Hall.
- Practical Unix & Internet Security (3rd Ed.), S. Garfinkel, G. Spafford, A. Schwartz, O'Reilly.
- Firewalls and Internet Security: Repelling the Wily Hacker (2nd Ed.), W. Cheswick, S. Bellovin, A. Rubin, Addison Wesley.

Riferimenti su Internet Security

- Building Internet Firewalls, E. Zwicky, S. Cooper, B. Chapman, O'Reilly.
- **The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd Ed.)**, D. Stuttard, M. Pinto, Wiley, 2011.
- **The Art of Software Security Assessment: Identifying And Preventing Software Vulnerabilities**, M. Dowd, J. McDonald, J. Schuh, Addison Wesley, 2006.
- **Serious Cryptography**, J.P. Aumasson, No Starch, 2018.