Sicurezza

1 - Introduzione

Ogni sistema sicuro deve avere i meccanismi per l'autenticazione, l'autorizzazione e la revisione. La sicurezza di un meccanismo non dovrebbe dipendere dall'ignoranza dell'attacker su come il meccanismo funziona o è costruito (no "sicurezza attraverso l'oscurità").

I principi fondamentali della computer-security sono:
Confidenzialità - si riferisce all'occultamento di informazioni o risorse. Un meccanismo di controllo degli accessi per preservare la confidenzialità è la *crittografia*.
Integrità - si riferisce all'affidabilità di dati o risorse, è normalmente espresso in termini di prevenzione di modifiche improprie o non autorizzate. Si divide in due classi:
- Meccanismi di *prevenzione* (meccanismi atti a mantenere l'integrità dei dati bloccando qualsiasi tentativo non autorizzato di modificare i dati o qualsiasi tentativo di modifica dei dati in modi non autorizzati)
- Meccanismi di *detection* (non provano a prevenire violazioni di integrità ma riportano semplicemente che l'integrità dei dati non è più affidabile)
Disponibilità - si riferisce alla possibilità di utilizzare le informazioni o le risorse desiderate.

Minaccia: potenziale violazione di sicurezza, alcuni esempi sono:
Snooping: intercettazione di informazioni non autorizzata.
Passive Wiretapping: forma di snooping nella quale è una rete ad essere monitorata.
Modification/Alteration: modifica di informazioni non autorizzata.
Active Wiretapping: alterazione di dati che si spostano attraverso una rete, es. man-in-the-middle - inganno.
Spoofing/Masquerading: impersonalizzazione di un'entità da parte di un'altra.
Ripudio dell'origine: falsa negazione dell'invio, o creazione, di un qualcosa da parte di una certa entità - inganno.
Negazione della ricezione: falsa negazione della ricezione di alcune informazioni o messaggi da parte di una certa entità - inganno.
Ritardo: temporanea inibizione di un servizio - usurpamento.
Negazione di un servizio: duratura inibizione di un servizio - usurpamento.

Def.: Una *politica di sicurezza* è una dichiarazione di cosa è, e non è, permesso.
Def.: Un *meccanismo di sicurezza* è un metodo, uno strumento, o una procedura che ha lo scopo di far rispettare una *politica di sicurezza*.

Gli *obiettivi* della sicurezza informatica sono:
Prevenzione - ha scopo di impedire un attacco, i meccanismi di prevenzione possono impedire la compromissione di parti del sistema.
Detection - utile quando un attacco non può essere prevenuto, può anche indicare l'efficacia delle misure di prevenzione. L'obiettivo è determinare che un attacco è in corso, o è avvenuto, e riportarlo.
Recupero - ha due forme: la prima è fermare un attacco, stimare e rimediare ai danni causati dall'attacco. La seconda forma ha lo scopo di mantenere la continuità del sistema durante l'attacco.

Sicurezza di un sistema:
Sia P l'insieme di tutti gli stati possibili. Sia Q l'insieme di tutti gli stati sicuri (secondo le politiche di sicurezza in atto). Sia R l'insieme degli stati resi disponibili dal sistema attuando i meccanismi di sicurezza (quindi $R \subseteq P$). Si ha la seguente definizione:

Def.: un meccanismo di sicurezza è *secure* se $R \subseteq Q$; è *precise* se $R = Q$; ed è *broad* se esistono stati r tali che $r \in R$ e $r \notin Q$.

Living Off the Land attack: attacco che sfrutta risorse esistenti e non malware o vulnerabilità (ad esempio phishing, strumenti off-the-shelf, servizi cloud, ecc).

Legacy Software: software vecchio ma mantenuto perché costerebbe troppo, in tempo e denaro, rinnovarlo.

Un componente COTS (Commercial-Off-the-Shelf), si riferisce a componenti hardware e software disponibili sul mercato per l'acquisto. Ad esempio Microsoft Windows, Adobe, Processori Intel,… Lo svantaggio è il minore livello di affidabilità (un prodotto customizzato è più affidabile).

Un *Network Information System* (NIS) comprende computer, comunicazione e persone (es. internet, sistema di telefonia mobile, difesa dai missili balistici).

Un NIS è affidabile quando funziona in modo corretto anche in caso di: attacchi malevoli/ostili, errori di implementazione e/o design (bugs), errori generati dagli utenti, interruzione dell'ambiente.

Definizioni
Vulnerabilità: debolezza in un sistema che può essere sfruttata per creare un danno
Attacco: una metodologia che sfrutta una vulnerabilità
Minaccia: un avversario motivato e capace che lancia un attacco
Window of Opportunity: intervallo temporale tra il primo attacco che sfrutta una vulnerabilità e l'applicazione della patch rilasciata dal venditore affetto.
Zero-day attack: un attacco che avviene durante la Window of Opportunity.

Le strategie di difesa sono essenzialmente due: identificare poi eliminare tutte le vulnerabilità nel sistema e identificare le minacce ed eliminare quelle vulnerabilità che le minacce sfruttano.

Esempi:
Heartbleed – vulnerabilità scoperta nel 2014, riguarda l'implementazione SSl del TSL, protocollo di comunicazione tra browser e server. In pochi giorni è stata rilasciata la patch per risolvere la vulnerabilità.

Shellshock – vulnerabilità scoperta il 25/09/2014, riguarda l'interprete della linea di comando nei sistemi Unix (GNU Bourne-Again Shell). Tale vulnerabilità è stata scoperta dopo 25 anni dalla sua creazione, in questo tempo si è diffusa in tantissimi sistemi (specialmente web-servers Linux).

2 - Crittografia

*Un **crittosistema** è una tupla di 5 elementi (E, D, M, K, C), dove M è l'insieme dei plaintexts, K l'insieme delle chiavi, C l'insieme dei ciphertexts, E : M x K→C l'insieme delle funzioni di cifratura, e D : C x K→M l'insieme delle funzioni di decifratura.*
Confidenzialità: comunicazione privata (segreta) in un ambiente pubblico.
Segretezza perfetta: confidenzialità sempre garantita.
Segretezza computazionale: confidenzialità garantita solo se le risorse dell'avversario sono limitate.

Poiché l'informazione è composta da un alfabeto finito (bits) e ha una lunghezza finita (n bits), confidenzialità perfetta è irraggiungibile. Un avversario può semplicemente "indovinare" i valori di tutti gli n bits con probabilità $1/2^n$ (tramite brute-force).

Cifrario Perfetto: se possiamo cifrare n bits di informazione tale che un avversario non possa risalire a suo contenuto con probabilità maggiore di $1/2^n$ utilizzando un metodo brute-force allora parliamo si cifrario perfetto.

Definizioni
Crittografia: progetto di cifrari sicuri ed efficienti

Crittoanalisi: metodi, strumenti e tecniche per attaccare i cifrari (valutare la loro bontà)
Crittologia: crittografia + crittoanalisi
Plaintext: input alla funzione di cifratura
Ciphertext: output dalla funzione di cifratura
Crittografia a chiave Privata/Segreta (simmetrica): le chiavi di cifratura e decifratura sono le stesse così come le funzioni di cifratura (C) e decifratura (D) sono spesso le stesse.
Crittografia a chiave Pubblica (asimmetrica): le chiavi (e le funzioni) di cifratura e decifratura sono diverse.
NOTA: le funzioni di cifratura/decifratura utilizzate in una conversazione sono conosciute da tutti (anche da un crittoanalista intruso), si può avere sicurezza ma non segretezza (degli strumenti usati per criptare).

Attacchi (conoscendo dell'algoritmo di crittografia usato):
- *Brute Force*: non ha bisogno di altro, prova tutte le combinazioni possibili.
- *Ciphertext only attack*: l'attacker ha bisogno solo del ciphertext, l'obiettivo è trovare il corrispondente plaintext (e se possibile, anche la chiave).
- *Known plaintext attack*: l'attacker conosce il ciphertext e il plaintext che è stato cifrato, l'obiettivo è trovare la chiave utilizzata.
- *Chosen plaintext attack*: l'attacker ha la possibilità di richiedere il ciphertext di specifici plaintexts, l'obiettivo è trovare la chiave utilizzata.

Funzioni
Cifratura - $\mathbf{C_k(m) = c}$ (cifratura di m con chiave k)
Decifratura - $\mathbf{D_k(c) = m}$ (decifratura di c con chiave k)
Matematicamente D_k è l'inversa di C_k:
$D_k(C_k(m)) = m$ [inoltre se $D_k(C_k(m)) = C_k(D_k(m)) = m$, allora ha proprietà commutativa]

Queste funzioni si possono dividere in due grandi famiglie:
Cifrari per uso *ristretto*: C e D sono tenute nascoste al mondo esterno, non è previsto quindi l'uso di una chiave segreta. Per un uso ristretto e specifico è attuabile per un uso generale no.
Cifrari per uso *generale*: C e D sono note a tutti, la sicurezza si basa sull'utilizzo di una chiave segreta nota solo al mittente o al destinatario.

Crittografia a chiave segreta (simmetrica)

È necessario accordarsi sulla chiave, infatti richiede l'uso di un canale "out-of-band" ritenuto più sicuro di quello tra A e B in quanto qui ci si scambierà la chiave. Tali canali tuttavia possono essere costosi e disponibili poco frequentemente (come effettuare lo scambio di persona). Inoltre lo spazio delle

chiavi deve essere molto grande (no brute-force attacks). Alcuni esempi di cifratura a chiave simmetrica: DES, Triple-DES, Blowfish, IDEA, AES.

Il ruolo di C_k e D_k è intercambiabile (spesso sono un'unica funzione). Il mittente e il destinatario conoscono la stessa chiave k, la segretezza delle chiavi dipende quindi da entrambi. I difetti sono che accordarsi sulla stessa chiave può essere complicato, per O(n) utenti bisogna accordarsi su $O(n^2)$ chiavi (ogni utente ha una chiave diversa con ogni altro), un numero troppo grande per permettere di memorizzare e scambiare chiavi segretamente.

Cifrari

Cifrari a trasposizione: riarrangia i caratteri nel plaintext per formare il ciphertext, lasciando le lettere invariate. Matematicamente la chiave è una funzione di permutazione, in quanto tale funzione non altera la frequenza dei caratteri.
Es. "HELLO WORLD"
HLOOL
ELWRD
Ciphertext: "HLOOLELWRD"
Questo cifrario può essere rotto rilevando la frequenza dei caratteri, e utilizzando crittoanalisi di tipo statistica.

Cifrari a sostituzione: modifica i caratteri nel plaintext per produrre il ciphertext. Un esempio è il cifrario di Cesare che prevede 26! possibili permutazioni dell'alfabeto (chiavi), un brute-force sarebbe impraticabile senza crittoanalisi di tipo *statistica* sulla frequenza delle lettere (o coppie di lettere).

Cifrari polialfabetici: Si sceglie una sequenza di chiavi, rappresentate da una stringa, le lettere della chiave sono applicate ai caratteri in successione del plaintext e quando si aggiunge la fine della chiave, la chiave riparte. La lunghezza della chiave è chiamata *periodo* del cifrario, questo richiede molte lettere chiave diverse. Nella pratica tramite una sorta di matrice ad ogni riga vi è una permutazione diversa dell'alfabeto, l'incontro riga-colonna determina la lettera che farà parte del ciphertext. Un attacco brute-force con anche statistica sulla frequenza è possibile ma decisamente più difficile. Si è scoperto che se vi sono blocchi di testo ripetuti della stessa lunghezza della chiave, anche nel ciphertext si avranno blocchi corrispondenti ripetuti.
Crittografia a chiave segreta (cifrari polialfabetici): invece di sostituire singole lettere del plaintext si sostituiscono blocchi di lettere, crittoanalisi di tipo statistico non è (quasi) più possibile.
Crittografia a chiave segreta (cifrari a permutazione): si cambia l'ordine delle lettere nel plaintext, i numeri nella chiave indicano le lettere (verticalmente) prese per il ciphertext. Questo procedimento può essere ripetuto più volte, si cifra

nuovamente il precedente ciphertext (un esempio di questa cifratura è la macchina Enigma).

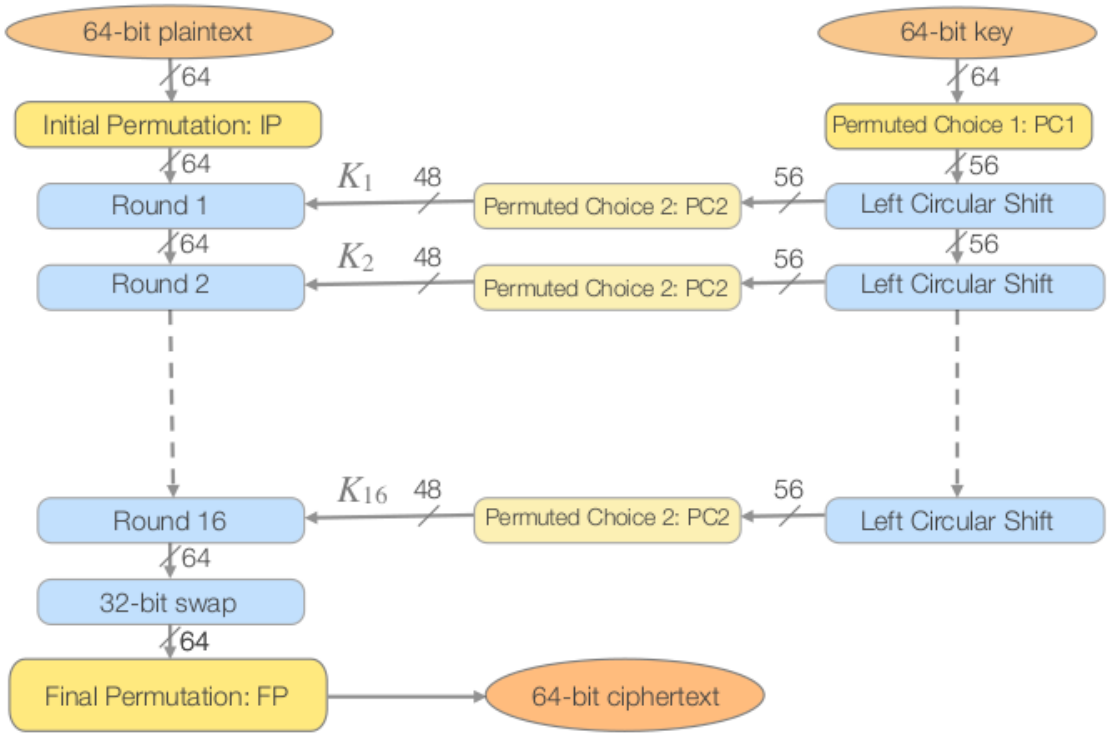
One-Time Pad (cifrario perfetto)

Ha una chiave e utilizza lo stesso algoritmo per cifrare e decifrare.
Utilizza una chiave lunga quanto il messaggio, creata in modo casuale e tramite uno XOR tra plaintext e chiave si ottiene il ciphertext.
Lo XOR è riproducibile tramite aritmetica modulare $c_j = m_j + k_j \bmod 2$ (mod 26 nel caso dell'alfabeto). La fase di decifratura è uguale, si effettua uno XOR tra ciphertext e chiave.

Pregi: se ogni bit della chiave è generato casualmente, allora conoscere un bit del ciphertext non fornisce nessuna informazione aggiuntiva sul corrispondente bit del plaintext, si parla quindi di cifrario perfetto (probabilità di trovare la soluzione $1/2^n$).
Difetti: la chiave è lunga quanto il messaggio (può essere estremamente lunga), si usa solo per quel messaggio (one-time) e occorre scambiarsela.

Data Encryption Standard (DES)

Il primo cifrario ad essere certificato 'robusto' da un ente nazionale (NSA – 1977).
È un cifrario simmetrico (a chiave segreta), suddivide il plaintext in blocchi da 64-bit (*non* è uno *stream cipher*: un cifrario in grado di cifrare bit man mano che

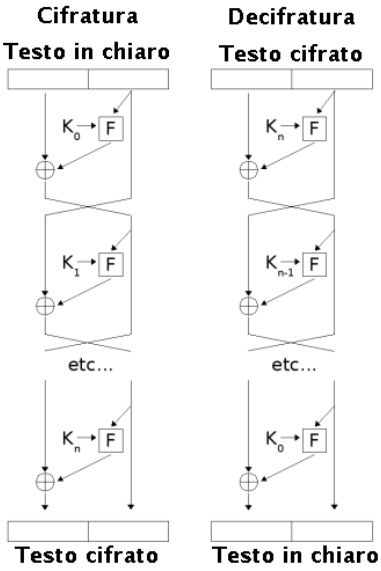


vengono prodotti) e necessita di una una chiave segreta (di 64-bit di cui 56 utilizzati e 8 come 'parity check'). Dato un plaintext di n bit restituisce un ciphertext di n bit.
Le operazioni utilizzate da questo cifrario sono: permutazione, sostituzione, espansione, contrazione e shift circolare (sx e dx).
NOTA: una chiave di 56-bit → spazio 2^{56} ad oggi è craccabile con un attacco brute-force, per questo ad oggi è non è più utilizzato.

Esecuzione
Brevemente
Questo algoritmo è bit-oriented, l'input, l'output e la chiave sono ognuna di 64-bits, tali insiemi di 64-bits sono chiamati *blocchi*. Il cifrario consiste di 16 rounds, ognuno dei quali utilizza una chiave separata di 48-bits. Queste round keys sono generate dal blocco della chiave eliminando i bits di parità (riducendolo a 56-bits), permutando i bit ed estraendo 48 bits. Un diverso insieme di 48 bit è estratto per ognuno dei 16 round. Se l'ordine nel quale le round keys sono utilizzate viene invertito, l'input è decifrato. I round sono eseguiti sequenzialmente, l'input di un round è l'output del precedente. La metà destra dell'input e la round key, sono "processate" da una funzione f che produce 32 bit di output, a tale output è applicato uno XOR con la metà sinistra e le risultanti metà destra e sinistra sono invertite. La funzione f indica la forza del DES. La metà destra dell'input (32-bits) è espansa in 48-bits, a questa viene poi applicato uno XOR con la round key. I 48 bit risultanti sono splittati in 8 insiemi di 6 bit ciascuno, ed a ogni insieme è applicata una tabella di sostituzione chiamata S-box, ogni S-box produce 4 bit di output. Sono poi concatenati in un insieme unico di 32-bits, il quale sarà infine permutato. I 32-bit costituiscono l'output della funzione f.

In dettaglio
Il DES è un algoritmo block cipher, ovvero opera su un gruppo di bit di lunghezza finita, che vengono organizzati in un blocco; tali algoritmi operano per iterazione. La chiave è inizialmente espansa in un insieme di 'round keys' e successivamente una 'round function' nella forma $R(k_{i...n}, m)$ è applicata al messaggio per tutti gli n round. È importante quindi definire il meccanismo di "espansione" della chiave e la funzione di round. Il DES che ha come lunghezza dei blocchi 64-bits e come lunghezza della chiave 56-bits.
Il DES utilizza il cifrario di Feistel, di seguito matematicamente:
Poniamo F essere la funzione dei passaggi e $K_0,K_1,...,K_n$ le sotto-chiavi rispettivamente dei passaggi 0,1,...,n. Le operazioni basilari sono dunque le seguenti:

Dividi i dati in ingresso in due parti uguali (L_0, R_0), Per ogni round i = 1,2,...,n calcola:
$R_i = L_{i-1}$ XOR f(R_{i-1}, K_{i-1})
$L_i = R_{i-1}$
dove f è la funzione del round e K_i è la chiave di sessione.
Si ottiene il testo cifrato (L_n,R_n).
Senza considerare la funzione f, la decifrazione si ottiene con
$R_{i-1} = L_i$
$L_{i-1} = R_i$ XOR f(L_i, K_i).



Cifrario di Feistel

Nel DES ci sono 16 round di cifrario di Feistel, ma prima di applicarli si parte da un blocco di 64-bits e si effettua un Initial Permutation (IP) che semplicemente scambia di posizione i bit secondo una certa tabella (disponibile online). Tale permutazione è presente anche alla fine, ma nell'esatto ordine opposto. Durante i 16 rounds la chiave di 56-bits (8-bits sono di validità) è espansa appunto in 16 'round keys' ognuna da 48-bits. Analizziamo la 'round function'.
Si prendono per prima cosa 32-bits del messaggio in chiaro e 48-bits della chiave i-esima. Dei 32-bits in chiaro si effettua un espansione (**E**), che non fa altro che replicare alcuni bit e spostarne altri producendo un output di 48-bits. A questo punto viene effettuato uno XOR con la 'round key', anch'essa di 48-bits, producendo, ovviamente, in output altri 48-bit. Questi vengono allora divisi in 8 sezioni da 6-bits l'una, ognuna delle quali viene "contratta" (**S**) in 4-bits, producendo un totale di 8*4 = 32-bits. Su questi viene effettuata una permutazione (**P**), scambiandoli di posizione e producendo 32-bits in output.
Approfondimento su **S-Box**: si hanno 6-bits da trasformare in 4. Questo avviene tramite una tabella prestabilita, nella quale nelle colonne sono presenti tutte le combinazioni possibili dei 4 bit centrali, e nelle righe tutte le combinazioni possibili dei 2 bit esterni 011011. Combinando riga con colonna si ottengono i 4 bit risultanti.

Crittografia a chiave pubblica (asimmetrica)

Permette lo scambio di informazioni segrete senza prima accordarsi sulla chiave. L'obiettivo è di rompere il legame tra C_k e D_k, chi sa cifrare non è in grado di decifrare. La chiave k (prima unica) ora viene vista come una coppia di chiavi pubblica e privata, dalla chiave pubblica non è (facilmente) ottenibile la chiave privata. Concetto di funzione "one-way trap-door", difficile andare dalla cifratura alla decifratura senza la chiave privata. Sono funzioni facili da calcolare ma difficili da invertire (a meno che non si conosca un elemento aggiuntivo, la chiave privata).

Funzioni One-Way con Trap-Door: Siano p, q due numeri primi:
- è facile calcolare n = p*q
- è difficile trovare p (o viceversa con q) dato n se non conosciamo q (p = n/q)

Rivest, Shamir and Adleman (RSA)

L'algoritmo RSA è uno dei primi sistemi di criptazione a chiave pubblica e si basa sulla teoria dei numeri primi.

Descrizione algoritmo
Sia Z l'insieme dei numeri interi, sia Z_n l'insieme dei numeri interi modulo n, sia Z^*_n l'insieme dei numeri interi primi con n, sia ϕ(n) la funzione di Eulero (= $|Z^*_n|$) e sia GCD(m, n) il massimo comun divisore tra m e n.
Dati p e q due numeri primi, allora:
ϕ(p) = (p - 1)
dato GCD(n, m) = 1 allora ϕ(mn) = ϕ(m)ϕ(n)
ϕ(pq) = (p - 1)(q - 1)

NOTE:
- Un numero è primo con un altro numero quando non hanno nessun divisore in comune diverso da 1, ad esempio 4 e 15 sono primi tra loro anche se non sono primi matematici.
- La funzione di Eulero associa a un numero intero *n* il numero dei numeri interi primi con *n* e minori di *n* (compreso l'uno)

Breve esempio: *n* = 15
Cerco numeri minori di 15 primi con 15: {1, 2, 4, 7, 8, 11, 13, 14}
Sono 8 numeri, quindi ϕ(15) = 8
Con Eulero: dato che 15 = 3 * 5
ϕ(15) = ϕ(3*5) = ϕ(3)*ϕ(5) = (3-1)*(5-1) = 2*4 = 8

Generazione delle chiavi
- Si scelgono due numeri primi (grandi): **p** e **q**
- Si calcola **n** = **p** * **q**
- Si sceglie **e** tale per cui GCD(**e**, ϕ(**n**)) = 1;
  ovvero ϕ(e*ϕ(n)) = ϕ(e) * ϕ(ϕ(n))
- Calcolo **d** tale per cui **e** * **d** mod ϕ(**n**) = 1
- Chiave pubblica: (e, n) - chiave privata: (d, n)

$C(m) = m^e \bmod n$
$D(c) = c^d \bmod n$
Considerazioni:
n ed **e** sono pubblici (disponibili a tutti), l'unica parte tenuta nascosta è **d**. Il messaggio **m** può essere di tipo arbitrario, quindi elevare un messaggio (di qualsiasi tipo) alla potenza **e** significa elevare la sua rappresentazione binaria.

Tale elevamento viene effettuato a blocchi di bit di lunghezza uguale. NOTA: Perché funzionino le funzioni di criptazione e decriptazione è necessario che le funzioni C e D siano inverse.

Esempio
Scegliamo **p** = 61, **q** = 53
Quindi n = 61*53 = 3233, $\phi(n) = (61-1)(53-1) = 3120$
Scegliamo e = 17 (in quanto GCD(e, $\phi(n)$) = 1)
Calcoliamo d = 2753 tale che e * d mod $\phi(n) = 1$
e * d = 2753 * 17 = 46801
e * d mod $\phi(n)$ = 46801 mod 3120 = 1
Le chiavi quindi sono: K[priv] = (2753, 3233) e K[pub] = (17, 3233)

Dimostrazione con K[priv] = (2753,3233) e K[pub] = (17,3233):
Sia m = 123, il messaggio in chiaro
Codifica = C(m) = m^e mod n = 123^{17} mod 3233 = 855 = c
Decodifica = D(c) = c^d mod n = 855^{2753} mod 3233 = 123 = m
NOTE: trattando grandi numeri elevati a potenze alte, si possono incontrare problemi di overflow.

Correttezza, Sicurezza ed Efficienza dell'RSA
È necessario dimostrare che D(C(m)) = m
Partiamo dal teorema di Eulero (già dimostrato), il quale dice che se GDC(m,n) = 1 allora $m^{\phi(n)}$ mod n = 1
Premessa (aritmetica modulare)
- se x mod n = 1, allora per qualsiasi intero y, x^y mod n = 1
- se x mod n = 0, allora per qualsiasi intero y, x^y mod n = 0

Dimostrazione

Sia m una codifica come intero del messaggio in chiaro tale che sia compresa tra 0 e n (dimensione di un blocco in cui divideremo il messaggio completo) → **0 < m < n**

Per definizione allora, si ha che:
1. D(C(m)) = D(m^e mod n)
2. D(m^e mod n) = (m^e mod n)^d mod n
3. (m^e mod n)^d mod n = (m^e)^d mod n [*applicata la proprietà modulare*]
4. (m^e)^d mod n = m^{ed} mod n
5. Per costruzione sappiamo che e*d mod $\phi(n) = 1$ [*garantito da **e** e **d** scelti opportunamente*]
6. Quindi deve esistere un intero positivo k tale che e*d = k * $\phi(n)$ + 1 [*per via della definizione di modulo*]
7. Sostituendo si ottiene:

$m^{ed} \bmod n = m^{k\phi(n)+1} \bmod n$
$m^{k\phi(n)+1} \bmod n = m^{mk\phi(n)} \bmod n$
$m^{mk\phi(n)} \bmod n = m$

Tale dimostrazione è veritiera se si verifica che è estendibile a qualsiasi m e non solo a quelli primi con n, come dice il teorema di Eulero (non verrà vista ma è così).
Da questa dimostrazione quindi si ottiene che RSA funziona con qualsiasi m messaggio (m).

Sicurezza RSA:
- Attacchi brute-force, per diminuire le probabilità di riuscita è consigliabile utilizzare uno spazio di chiavi sufficientemente ampio.
- Attacchi matematici: fattorizzare n per risalire a p e q oppure calcolare $\phi(n)$ senza fattorizzare n e successivamente calcolare $d = e^{-1} (\text{mod } \phi(n))$ (inverso modulare, non divisione intera). Utilizzando p e q sufficientemente grandi il costo di fattorizzazione di n diventa estremamente alto, n può essere composto anche da 600-bits.

La sicurezza di RSA è garantita finché la fattorizzazione di un intero è molto costosa (e impossibile nel breve tempo).

Efficienza RSA:
Come calcolare $x^y \bmod n$ efficientemente, con *y* potenza di 2
$x^{32} = x \rightarrow x^2 \rightarrow \dots \rightarrow x^{32}$
Avremo 5 moltiplicazioni in tutto perchè $5 = \log_2(32)$

Se y non potenza di 2:
Da x^y possiamo ottenere x^{2y} e x^{y+1} con solo una moltiplicazione per ognuno: $x^{2y} = x^y * x^y \mid x^{y+1} = x^y * x$

Ad esempio: $123^{54} \bmod 3233$
L'esponente in binario è $54 = 110110_2$
È quindi necessario calcolare $1284^{110110} \bmod 3233$

L'efficienza (supponendo di dover calcolare 123^{110110})
1284^{1} 1284
1284^{11} $1284^2 * 1284$
1284^{110} $(1284^2 * 1284)^2$
1284^{1101} $((1284^2 * 1284)^2)^2 * 1284$
…
1284^{110110}

Invece che calcolare 54 moltiplicazioni, aggiungendo man mano un bit al massimo vengono effettuate 12 moltiplicazioni, in quanto è $2 * \text{round}(\log_2(54))$

[2 * numero di bit del numero 54]. Risulta comunque molto complesso e poco efficiente, quindi sicuro.

NOTA: $\log_2(n)$ indica il numero di bit necessari a rappresentare n

Una proprietà dell'aritmetica modulare è:
$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$
Che fa si che ognuno dei risultati intermedi possa essere ridotto ad un modulo di n, infatti il risultato di un qualsiasi numero modulo n non potrà mai superare il valore n. Questo rende la computazione più pratica.

Generare numeri primi molto grandi:
Non essendo possibile consultare una tabella (in quanto i numeri che cerchiamo sono troppo alti). Scegliendo un numero dispari casuale, la probabilità che esso sia primo è 1/log(n).
Esempio, se prendessi n con 10 cifre la probabilità che sia primo è 1/23, se n avesse 100 cifre la probabilità sarebbe 1/230. Tali probabilità sono troppo piccole per utilizzarlo nell'RSA.

Supponiamo di avere una funzione p_test(n) che restituisce true se n è primo e false in caso contrario.
Un approccio naive è controllare che ogni intero k da 2 ad n-1 divide n. Un altro approccio migliore è invece testare tutti gli interi invece che fino ad n-1 fino a √n. La complessità rimane però O(√n) ordine di n, comunque troppo elevato. Nel 2002 è stato trovato un metodo con complessità $O((\log n)^6)$ per il test di primalità che abbassa di molto la complessità.

Algoritmo probabilistico per il test di primalità:
Partiamo dal piccolo teorema di Fermat: se n è primo allora per qualsiasi intero **a** tale che 0 < **a** < n allora: $\mathbf{a^{(n-1)} \bmod n = 1}$
Pomerance dice inoltre che: dato n un numero grande (> 100 cifre), e sia a un numero casuale positivo inferiore ad n, allora
Probabilità che $[(n \text{ non è primo}) \text{ e } (a^{(n-1)} \bmod n = 1)] \simeq 10^{-13}$

$a^{(n-1)} \bmod n \mathrel{!}= 1$ è sicuramente non primo, se invece è uguale a 1 vi è una probabilità 10^{-13} che però non lo sia effettivamente.
Per ridurre la possibilità di errore in caso fosse uguale ad 1 si potrebbe ripetere il test k volte con valori diversi di a ogni volta. In tal caso la probabilità che n non sia effettivamente primo si riduce a 10^{-13k}. Mediamente sono testati k = log(n)/2 numeri prima che si possa accettare un numero n come primo. Ad esempio prendendo un numero casuale di 200-bits saranno necessari $\log(2^{200})/2 = 70$ tentativi.

Checksum crittografico

Una funzione di checksum è una funzione matematica che genera un insieme più piccolo di k bit dagli n bit originari. Questo insieme più piccolo è chiamato checksum o message digest. Si invia poi al destinatario sia il messaggio che il checksum in modo che quest'ultimo possa ricalcolare il checksum e confrontarlo con quello del mittente, se combaciano non vi sono state modifiche. Queste sono un esempio di funzioni hash.

Definizione
Una funzione crittografica di checksum (o funzione hash forte) $h:A \rightarrow B$ è una funzione con le seguenti proprietà:
1. Per ogni $x \in A$, $h(x)$ è facile da calcolare
2. Per ogni $y \in B$, è computazionalmente infattibile trovare $x \in A$ tale che $h(x) = y$
3. È computazionalmente infattibile trovare $x, x' \in A$, tali che $x \neq x'$ e $h(x) = h(x')$ [tale coppia è chiamata *collisione*]
4. Data una qualsiasi $x \in A$, è computazionalmente infattibile trovare un altro $x' \in A$ tale che $x \neq x'$ e $h(x') = h(x)$

Funzioni Hash

Una funzione da un dominio X ad uno Y, con la particolarità che il dominio X ha dimensione n mentre il domino Y ha dimensione m, la particolarità è che n è estremamente inferiore ad m.
$f : X \longmapsto Y$
$| X | = n, | Y | = m, \mathbf{n \gg m}$
dato $x \in X, y = f(x) \in Y$

Proprietà delle funzioni Hash
f deve essere una funzione molti-ad-uno ma "bilanciata", ovvero se prendo tutti gli x_i cha abbiano un digest simile ad y_i, non vi è una discriminazione quando si fa una mappatura verso certi digest invece che altri, la mappatura è uniforme verso tutti i digest possibili. La funzione f deve essere tale che valori molto vicini tra loro in X sono mappati molto distanti nel dominio Y.

Le funzioni Hash Crittografiche
Dette anche funzioni OneWay, sono funzioni che rispettano altre seguenti tre proprietà oltre alle normali funzioni hash:
1. Dato un $x \in X$, dev'essere facile calcolare $f(x)$
2. Dato un $y \in Y$, dovrebbe essere computazionalmente non praticabile trovare il rispettivo $x \in X$ tale che $f(x) = y$
3. Dato un qualsiasi x_1, deve essere computazionalmente non praticabile trovare un altro x_2 diverso da x_1 tale che $f(x_1) = f(x_2)$

Le proprietà di *sicurezza* sono:
Hiding: proprietà 2
Collision-freedom: proprietà 3

Dato un digest trovare un pre-digest è possibile con forza bruta calcolando tutte le possibili combinazioni di pre-digest finché non avviene il risultato desiderato. Quello che è possibile fare è rendere questa operazione inapplicabile in tempi decenti con le risorse attuali, aumentando la dimensione di x.
Implicazioni del collision-freedom:
Dati due digests che sono uguali $f(x_1) = f(x_2)$ allora è sicuro assumere che i pre-digests sono uguali $x_1 = x_2$ grazie alla proprietà di *Collision Freedom*.

Autenticazione Firma Digitale

Oltre alla *confidenzialità* la crittografia fornisce nuove funzionalità, come:
Integrità: il destinatario di un messaggio deve essere in grado di accertare che il contenuto del messaggio corrisponda a quello inviato dal mittente (checksum).
Autenticazione: il destinatario di un messaggio deve essere in grado di accertare l'identità del mittente del messaggio.
Firma digitale: funzionalità complessa richiesta quando mittente e destinatario di un messaggio non si fidano l'uno dell'altro (reciprocamente) - ha proprietà simili alla firma cartacea.

Proprietà della firma digitale
- non-falsificabile: a prova che il firmatario, e nessun altro, abbia deliberatamente firmato il documento.
- non-riutilizzabile: la firma è parte del documento e non può essere trasferita ad un altro.
- inalterabile: dopo essere stato firmato, il documento non può essere alterato.
- non-ripudiabile: il firmatario non può dire di non aver firmato il documento.

Una firma ammette due operazioni: una di firma del documento e una di verifica dell firma.

La firma digitale con crittografia asimmetrica (si firma con la chiave privata del mittente e di decripta con la sua pubblica) ha alcuni *punti a sfavore*: occorre un cifrario commutativo (come RSA), il documento firmato non è indirizzato ad uno specifico destinatario infatti tutti possono fare la verifica, non è confidenziale e la lunghezza del messaggio scambiato è il doppio della lunghezza del messaggio originario.

Una soluzione potrebbe essere criptare il messaggio con la chiave pubblica del destinatario, firmarlo e inviarlo.

Anche questa soluzione ha *punti a sfavore:* infatti non è possibile indicare se il messaggio ha senso o no (non è interpretabile).

È possibile allora prima firmare il messaggio, poi crittare il tutto con la chiave pubblica del destinatario, il quale decripterà con la propria privata. Questa soluzione è la migliore.

Message Authentication Code (MAC)

È un'immagine breve (digest) di lunghezza fissa del messaggio che può essere generata da un solo mittente conosciuto dal destinatario.
Può essere usata per autenticare il mittente e verificare l'integrità del messaggio, è ottenuto attraverso una funziona hash one-way e una chiave segreta condivisa tra il mittente e il destinatario.

Key Management

Vi sono vari problemi nella gestione delle chiavi necessarie al corretto utilizzo della crittografia.
Prima di tutto la distribuzione delle chiavi pubbliche, un primo approccio classico è l'*annuncio pubblico*, ovvero la pubblicazione della chiave pubblica ad esempio in fondo alla propria mail, o sulla homepage del sito. Questo approccio però è soggetto ad attacchi di tipo man-in-the-middle, in cui si può intercettare la mail/documento html e sostituire la chiave pubblica. Un'altra modalità potrebbe essere l'*elenco pubblico*, dove le chiavi sono tenute da un'entità fidata in cui è presente un accesso con autenticazione. Anche qui vi sono problemi come la necessità di un entità fidata e la possibile violazione della directory contenente le chiavi pubbliche. Infine si potrebbero sfruttare dei *certificati*, con i quali un'autorità aggiunge la propria firma alla chiave garantendone l'autenticità.
La gestione delle chiavi private è un altro problema, considerando innanzitutto che per permettere ad n soggetti di comunicare tra loro sono necessarie n^2 chiavi private.
Assumendo che un Key Distribution Server (KDS) fidato condivida una diversa chiave segreta con ogni utente e A e B vogliono stabilire una comunicazione sicura. Il KDS genera e distribuisce ad A e B una chiave one-time session per essere utilizzata per la durata della comunicazione, le comunicazioni future tra A e B necessiteranno di differenti session keys. Descrizione del protocollo: la premessa è che A e B condividono con il KDS rispettivamente le chiavi K_A e K_B con cui può criptare cose che verranno decriptate solo dai rispettivi padroni delle chiavi. Prima di tutto A invia al KDS una richiesta di session key per B, il KDS genera una nuova chiave di sessione K_S e la invia ad A insieme alla K_S criptata con K_B (il messaggio di risposta del KDS è criptato con K_A). A decripta il messaggio e memorizza K_S, invia poi a B la K_S criptata con K_B, B decripta K_S e la memorizza; ora A può inviare messaggi criptati a B utilizzando la K_S. I problemi di

questo protocollo sono che B non può effettivamente sapere che il messaggio arriva da A. Questo problema ("replay attacks") viene risolto dal protocollo Needham-Schroeder che è del tutto simile al precedente ma prevede una challenge-response tra B ed A una volta che il primo riceve la K_S criptata da A.

Kerberos
I client (principals) provano la propria identità ai server senza l'invio di dati attraverso la rete che permetterebbe ad un attacker di rubarne l'identità. Ogni principal inizialmente condivide una chiave segreta (password) con il KDS, per ridurre l'esposizione di quest'ultima, il KDS ne utilizza una diversa per ogni sessione login. Tutte le comunicazioni all'interno di una singola sessione sono rese sicure grazie a chiavi ottenute da un Ticket Granting Server (TGS).
Se A vuole comunicare con un diverso principal C, deve far ripartire il protocollo con TGS (non KDS) per generare una nuova chiave di sessione K_{AC} utilizzando la chiave K_S (non la chiave segreta condivisa K_A)

In un sistema estremamente grande, KDS potrebbe generare un collo di bottiglia sull'affidabilità. Il KDS può essere replicato per ottenere un incremento di performance e affidabilità utilizzando uno schema master-slave. In un sistema esteso, un singolo (o replicato) KDS può non essere accettato per ragioni amministrative.
Lo schema con KDS riduce da $O(n^2)$ a $O(n)$ il numero di chiavi segrete per permettere a n utenti di comunicare con tutti, tuttavia richiede la presenza di un KDS sicuro.

Lo schema master-slave permette che solo il master possa cambiare il contenuto, mentre tutti gli altri possono solo leggere, è possibile un collo di bottiglia quando in scrittura tutto deve passare dal "master" (tuttavia tali operazioni sono molto più rare). Le chiavi sono generate al momento in cui l'utente effettua il login, la chiave segreta tra l'utente e il KDS è la user-password.

Gestione delle chiavi segrete tramite soluzioni ibride: un sistema crittografico ibrido utilizza sia crittografia simmetrica che asimmetrica. In generale la crittografia a chiave pubblica è relativamente lenta (circa 1000 volte rispetto a crittografia a chiave privata), quindi per velocizzare, una volta distribuite le chiavi pubbliche, le uso per generare le chiavi segrete. Si utilizza la crittografia a chiave pubblica in una fase iniziale per scambiarsi una chiave segreta, poi si passa alla crittografia a chiave privata (usando la chiave segreta appena condivisa) per le fasi successive.

Funzionamento (lo stesso di **SSL**: https):
1. A: genera (K_A[pub], K_A[priv])
2. A: invia a B la propria K_A[pub]

3. B: genera K_S
4. B: invia ad A K_S criptata con K_A[pub]
5. A: decripta il messaggio ottenuto da B con la propria K_A[priv]
6. Eliminazione di (K_A[pub], K_A[priv])
7. A e B iniziano ad utilizzare la crittografia a chiave simmetrica utilizzando la chiave di sessione K_S

Lo schema ibrido è molto funzionale e più veloce di una crittografia sempre a chiave pubblica o sempre a chiave privata, anche se rimane comunque il problema del man-in-the-middle.

Certificati e Public-Key Infrastructures

I certificati digitali vengono utilizzati per evitare che qualcuno tenti di "spacciarsi" per un'altra persona sostituendone la chiave pubblica (lo stesso ruolo che ha una carta d'idendità rilasciata da un ente comunale). *Un certificato è un token che lega un'identità ad una chiave crittografica.*

Un *certificato digitale*, nella crittografia asimmetrica, è la forma nella quale le chiavi pubbliche sono comunicate. Tale certificato è un abbinamento tra una chiave pubblica e le informazioni identificative di un soggetto, firmato da un autorità. Ha lo scopo di evitare attacchi man-in-the-middle (evita la sostituzione della chiave pubblica senza modificare i soggetti), in quanto per modificare un campo si deve ricreare un documento con le varie firme: impossibile in quanto una firma digitale è unica.

X.509 (the Directory Authentication Framework)

L'X.509 è uno standard che specifica certificati digitali con i seguenti campi: subject (oggetto in questione), issuer (chi ha rilasciato il certificato), validity interval (le date di inizio e di fine), administrative info, extended info,…

La *distribuzione* dei certificati avviene tramite entità fidate come Certificate Servers e/o Public Key Infrastructures (PKI).

I *Certificate Servers* sono database centralizzati disponibili in rete, tali db possono essere però replicati. Permettono agli utenti di richiedere l'inserimento del proprio certificato nel database e richiedere il certificato di qualcuno

Una *Certification Authority* (CA) è un entità responsabile della certificazione, validazione e revocazione dei certificati. Vi sono molti tipi diversi di CAs tra cui commerciali, governativi e gratuiti, alcuni esempi sono VeriSign, Symantec, Thawte, Geotrust, Comodo, Visa, Actalis. Due CAs sono cross-certified se ognuna ha fornito un certificato all'altra.

Una *Public Key Infrastructure* (PKI) è una collezione di CAs e protocolli che permettono ai clienti di invocarne i servizi. Può essere vista come una rete che gestisce il rilascio di un certificato e ne controlla l'autenticità.

Come avviene la *certificazione* da parte di un PKI: prima di tutto quando qualcuno richiede un certificato per la prima volta, il soggetto deve essere autenticato, questo processo può avvenire tramite *autenticazione in-band* (utilizza solo le risorse presenti nel PKI ed è possibile solo per certi tipi di informazioni identificative come email, indirizzi, ecc) oppure autenticazione *out-of-band* (effettuata tramite metodi più tradizionali, come email, fax, tramite telefono, o anche di persona).

Il processo di *certificazione* è basato sulla fiducia: gli utenti si affidano all'autorità per fornire solo certificati che associano correttamente soggetti alle loro chiavi pubbliche. Ovviamente non c'è un solo CA per tutto il mondo, invece è comune che molti PKI abilitino un CA per certificare altri CA. Un CA dice agli utenti che possono fidarsi di ciò che un'altro CA dice nei propri certificati. Si parla quindi di "*Certificate Chains*".

Le *Certificate Chains* si basano sull'uso di certificati emessi dalle parti stesse e sull'attestazione della bontà di questi certificati da parte di terzi. Ogni certificato nella catena è validato dal precedente, i CA possono quindi essere organizzati come un albero radicato (es. X.509) oppure come un grafo generale (es. PGP).

Esempi di implementazione di questo schema sono GPG (*GNU Privacy Guard*) e PGP (*Pretty Good Privacy*). Grazie alla diffusione di PGP (e delle sue varianti) in congiunzione con l'e-mail, la rete di fiducia creata originariamente da PGP è la più estesa PKI bi-direzionale esistente (dati del 2004). CAcert.org gestisce una rete di fiducia simile a quelle usate da PGP, con la differenza che le informazioni relative alle relazioni di fiducia tra le parti vengono mantenute su database centralizzati.

Fiducia Gerarchica (X.509)
Si basa su catene di fiducia che formano un albero radicato tra le entità che sono reputate essere CAs, la fiducia (cieca) che affidiamo al livello radice dei CAs deve essere acquisita tramite reputazione, esperienza, competenza operativa e altri aspetti non tecnici. Chiunque voglia essere un CA dev'essere un'entità fidata e dobbiamo credere sia sicura e corretta.

Web of Trust (PGP)
In PGP, ogni utente può comportarsi come una CA e firmare la chiave pubblica di un altro utente, una chiave pubblica è infatti considerata valida solo se un sufficiente numero di utenti fidati l'ha firmata.
Man mano che il sistema si evolve, emergono complesse relazioni di fiducia come "rete" dinamica, la fiducia è necessario che non sia simmetrica o transitiva.

Per la *validazione* in PKI è necessario controllare che il certificato sia: *attuale* (tra le date "not before" e "not after"), *valido* (firmato da una catena di CAs che terminano in una radice CA) e *non revocato*. Il controllo dell'attualità e della validità può essere fatto localmente e off-line dall'utente certificato, il controllo di revoca del certificato è molto più complesso.

La *revocazione* è il processo di rottura dell'unione tra la chiave pubblica e un soggetto per varie ragioni, come ad esempio il caso in cui la chiave privata del soggetto viene compromessa o le informazioni identificative del soggetto cambiano (es. nome, indirizzo email,...). On-line la revocazione è semplice, l'*Online Certificate Status Protocol* (OCSP) dell'X.509 descrive come controllare la validità e revocare un certificato. Off-line invece finché il certificato è attuale, il metodo di revocazione è critico, un'alternativa può essere che i clienti controllino "localmente" se un certificato è stato revocato.

La Certificate Revocation List (CRL) è una lista di certificati revocati costruita, firmata e periodicamente distribuita dal CA, l'utente deve controllare l'ultimo CRL durante la validazione per assicurarsi che un certificato non sia stato revocato. X.509 include un profilo CRL, che descrive il formato dei CRLs. Alcuni problemi però possono essere la periodicità di aggiornamento delle CRL o la loro dimensione.

NOTA: il fatto di essere radice significa che ci si può auto-certificare.

Key Escrow

In molte situazioni una chiave (segreta) è conosciuta ad un solo individuo (es. chiavi per crittografia simmetrica, master-password per i password managers, ecc..). Cosa succede se si dimentica la chiave (segreta) o non è più accessibile, ad esempio in caso di morte?

Escow (definizione dizionario): un accordo contrattuale in cui una terza parte neutrale riceve e distribuisce soldi o documenti per le parti primarie della transazione, con l'erogazione dipendente dalle condizioni concordate dalle parti della transazione.
Es. il ruolo di PayPal in una transazione su eBay ad esempio, tra due parti non fidate.

Key Escrow: un accordo in cui le chiavi private sono "mantenute in garanzia" in modo che sotto certe circostanze, una terza parte autorizzata possa averne accesso.

Secret Sharing

Si splitta la chiave segreta in n parti, e se ne invia ognuno a un diverso ufficio sicurezza (o si cripta con l'i-esima parte della chiave pubblica dell'i-esimo ufficio sicurezza e mantenuto su disco). Nessun sottoinsieme degli uffici sicurezza sarà in grado di ricostruire la chiave, solo tutti gli n uffici sicurezza insieme, collaborando sono in grado di farlo.

Una soluzione naive potrebbe essere dividere semplicemente la chiave segreta S che è lunga k bit in n segmenti di k/n bit ognuno (eccetto l'ultimo). Tuttavia questa soluzione non soddisfa la richiesta "nessun sottoinsieme proprio degli uffici sicurezza è in grado di ricostruire la chiave in modo più facile di un attacco brute-force". Ogni ufficio sicurezza infatti conosce esattamente k/n bits della chiave segreta
Ogni sottoinsieme di m<n uffici sicurezza può portare a termine un attacco brute-force sui bit rimanenti, in questo modo cala drasticamente il numero di possibili combinazioni all'aumentare del sottoinsieme degli uffici sicurezza che si riuniscono.

2-Way Secret Sharing

Si splitta la chiave segreta S in due uffici sicurezza tali che: (confidenzialità) nessun ufficio sicurezza da solo ha alcuna informazione riguardo ad S, (disponibilità) i due uffici insieme possono ricostruire S.

Il funzionamento è semplice:
- Sia S una chiave segreta di k bits e n = 2
- Si genera una stringa random R di lunghezza k
- Si affida all'ufficio sicurezza uno la stringa $S_1 = R$
- Si affida all'ufficio sicurezza due la stringa $S_2 = R \oplus S$
- Per ricostruire la chiave segreta è necessario effettuare $S_1 \oplus S_2$

n-Way Secret Sharing

E' una generalizzazione di n, si splitta la chiave segreta S in n uffici sicurezza tali che (confidenzialità) nessun ufficio sicurezza da solo ha alcuna informazione riguardo ad S, (disponibilità) gli n uffici insieme possono ricostruire S.

Funzionamento:
- Sia S una chiave segreta di k bits
- Si generano n-1 stringhe random $R_1,R_2,...,R_n$ di k bits ognuna
- Si affidano le prime n-1 stringhe a n-1 uffici sicurezza
- Si affida l'ultima (n-esima) stringa ad un ufficio sicurezza nella forma: $R_1 \oplus R_2 \oplus \dots \oplus R_{n-1} \oplus S$
- Per ricostruire la chiave segreta è necessario
$S_1 \oplus S_2 \oplus \dots \oplus S_{n-1} \oplus S$

Schemi Threshold

Sorge un problema nel caso in cui uno degli uffici con cui è condivisa la chiave non sia (mai) più reperibile, in tal caso i restanti pezzi non sono in grado di recuperare l'intera chiave segreta. È necessario rendere meno rigida la richiesta che tutti gli n pezzi siano presenti per ricostruire la chiave.
Lo schema di threshold (t, n) [dove $t \leq n$]: viene splittata la chiave segreta in n "porzioni", t (o più) porzioni sono sufficienti per recuperare la chiave, con meno di t porzioni non si è in grado di recuperare la chiave.

Condivisione della chiave segreta - threshold (t, n):
Si genera un polinomio casuale di grado t-1 del tipo:
$g(x) = (a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \dots + a_1x + S) \bmod p$
[dove S è la chiave segreta e p un numero primo più grande di ognuno dei coefficienti, che è reso pubblico]. Si generano poi n porzioni, tali che $S_1 = g(1)$, $S_2 = g(2)$,..., $S_n = g(n)$. Si scarta successivamente il polinomio (i coefficienti ed S), e ora ogni porzione di t elementi è sufficiente per risolvere le t incognite in modo da ottenere il polinomio iniziale (e la chiave S).

Esempio:
Considerato l'esempio con schema threshold (2, n)
Il grado del polinomio è (t-1) = (2-1) = 1
$g(x) = (ax + S) \bmod p$ [lo schema indica al variare di g(x) come cambia la chiave segreta S]. Sono dati due punti (2 porzioni), in quanto sono necessari almeno due punti per definire una retta.

Dato invece un punto solo (una sola porzione), possono passarvi infinite rette al suo interno, in questo modo non si sa quale retta sia stata usata per definire la chiave S. In questo modo (in caso di t grande) più porzioni della chiave si conoscono e più è facile risalire ad essa. L'esempio mostra infatti che con una sola porzione (un punto) potrebbero esserci infinite chiave (infinite rette), ma se si avesse anche un secondo punto (in quanto t=2) la chiave è unica (in quanto due punti collegati formano una retta precisa).

Esempio (numerico)
n = 3, t =2 → (2, 3) - dati S = 6, p =17
Si genera un polinomio casuale di grado t - 1 = 2 - 1 = 1
g(x) = (4x + 6) mod 17

Si calcolano 3 porzioni da dare a 3 uffici sicurezza:
g(1) = (4 + 6) mod 17 = 10
g(2) = (8 + 6) mod 17 = 14
g(3) = (12 + 6) mod 17 = 1

Come possono 2 uffici sicurezza, ad esempio 1 e 2, ricostruire la chiave segreta?
- Sanno che il polinomio ha la forma g(x) = (ax + S) mod 17
- Date le porzioni g(1) = 10 e g(2) = 14, hanno:
g(1) = (a + S) mod 17 = 10
g(2) = (2a + S) mod 17 = 14
Risolvendo il sistema di equazioni per l'incognita S, ottengono 6 che è la chiave segreta iniziale.

PGP (Pretty Good Privacy)

PGP è una famiglia di software di crittografia per autenticazione e privacy. PGP usa sia la crittografia asimmetrica (a chiave pubblica) sia quella simmetrica (a chiave segreta), e fino ad un certo punto utilizza una Public Key Infrastructure (PKI) che ha qualche somiglianza con l'X.509. PGP usa la crittografia a chiave asimmetrica, con la chiave pubblica del destinatario il mittente cifra una chiave di sessione per un algoritmo di crittografia simmetrica; questa chiave viene quindi usata per cifrare il testo in chiaro del messaggio. Molte chiavi pubbliche di utenti PGP sono a disposizione di tutti dai numerosi Key Server PGP sparsi per il mondo, che operano come mirror reciproci.
Il destinatario di un messaggio protetto da PGP decifra prima la chiave di sessione inclusa nel messaggio usando la sua chiave privata. Poi decifra il testo usando la chiave di sessione con l'algoritmo simmetrico. L'uso di due cifrature è giustificato dalla notevole differenza nella velocità di esecuzione tra una cifratura a chiave asimmetrica ed una a chiave simmetrica (l'ultima è indicativamente 1000 volte più veloce). Ci sono inoltre delle vulnerabilità crittografiche nell'algoritmo a chiave asimmetrica utilizzato da PGP quando viene usato per cifrare direttamente un messaggio.
Una strategia simile può essere usata per capire se un messaggio è stato alterato, o se è stato mandato effettivamente da chi dice di essere il mittente. Per fare entrambe le cose, il mittente usa PGP per 'firmare' il messaggio con l'algoritmo di firma RSA o DSA. Per fare questo, PGP calcola un 'hash' (chiamato message digest) dal testo in chiaro, e crea poi da questo hash la firma digitale usando la chiave privata del mittente. Il destinatario del messaggio calcola il message digest dal testo in chiaro decifrato, e poi usa la chiave pubblica del

mittente ed il valore del message digest firmato con l'algoritmo di firma. Se la firma corrisponde al message digest del testo in chiaro ricevuto, si presuppone (con un grande margine di sicurezza) che il messaggio ricevuto non sia stato alterato né accidentalmente né volontariamente da quando è stato firmato. Questa proprietà è chiamata non ripudio: non si può ripudiare la paternità del messaggio.

PGP Keyrings
PGP memorizza le chiavi in due file chiamati keyrings, uno per le chiavi pubbliche (public keyring) ed uno per le chiavi private (user keyring). Le chiavi private sono salvate in forma cifrata utilizzando l'hash di una passphrase come chiave segreta.

Compressione
PGP utilizza inoltre la compressione per nascondere ulteriormente il messaggio codificato. Prima di criptare, il plaintext è compresso, dopo la decriptazione, il messaggio risultante è decompresso.

Alcuni algoritmi utilizzati in PGP sono: (chiave privata) CAST, Triple-DES, IDEA, AES, (chiave pubblica) RSA, ElGamal, DSA, (hash) SHA1

Certificati PGP
I certificati PGP contengono campi diversi da X.509 come: numero di versione PGP, la chiave pubblica di U, caratteristiche della chiave (lunghezza, algoritmo con cui è stata creata, data di creazione, durata della chiave,...), informazioni sull'identità di U (nome, cognome, luogo e data di nascita, foto,...), self-signature (chiave pubblica di U firmata con la chiave privata di U), indicazione dell'algoritmo simmetrico di codifica preferito (Es: CAST, IDEA, Triple-DES), altre firme...

È un certificato interoperabile, infatti riconosce i formati di certificato PGP, ma anche quelli in formato X.509. Alcune differenze: non è necessaria un'autorità (l'inventore, Zimmermann voleva infatti evitare che autorità statali, e non solo, intervenissero nella certificazione). PGP permette di abbinare una chiave pubblica a più identità a differenza di X.509 che abbia un certificato ad una singola identità. In PGP sono infatti presenti più firme per attestare la validità di un certificato, mentre in X.509 solo una. PGP può contenere molteplici coppie chiave/identità, ognuna firmata più volte.

Fiducia e Validazione
A meno che non ricevi un certificato direttamente dal possessore, è necessario affidarsi a qualcun'altro che è considerato valido. "Tu ti fidi delle persone, PGP valida i certificati". I modelli di fiducia possono essere: a fiducia diretta, a fiducia gerarchica o a rete di fiducia.

La fiducia in X.509 si basa su catene di fiducia tra entità reputate essere CAs. La fiducia cieca che poniamo nel livello radice dei CAs deve essere acquisita tramite la reputazione, l'esperienza, la competenza operativa e altri aspetti non tecnici.

La fiducia in PGP permette che ogni utente possa comportarsi come un CA e firmare il certificato di un altro utente (diventa un "introducer" per quella chiave). Si considera un certificato valido se e solo se ti fidi sufficientemente di uno o più degli introducers del certificato. Si assegna ad ogni soggetto, uno dei seguenti livelli di fiducia: completa fiducia, fiducia marginale o non affidabile. Se si assegna completa fiducia ad un soggetto diventa un CA.
La validazione in PGP considera un certificato "valido" basandosi su come gli altri soggetti giudicano il certificato e il livello di fiducia che si ha assegnato a quei soggetti. Si applica la propria firma solo su quei certificati che puoi verificare personalmente. Questo crea un "sistema di reputazione" dinamico e decentralizzato.
PGP considererà un certificato valido solo se, o viene apposta una firma da un utente completamente fidato, o vengono apposte due (o più) firme da utenti fidati marginalmente.
SDAs e PGP Shredder
Un Self-Decrypting Archive (SDA) è un archivio che può essere aperto da tutti, anche chi non ha PGP installato. Tali archivi possono essere protetti da pass-phrases.
PGP Shredder è uno strumento per cancellare i file in modo sicuro.

Secure Socket Layer (SSL)

Internet è strutturata a strati, più il livello è alto e maggiore è l'astrazione, la sicurezza di tipo SSL viene inserita tra il livello di trasporto e quello applicativo. SSL è un software fatto di vari moduli che implementano vari algoritmi di cifratura, in particolare utilizza un cifrario ibrido (simmetrici e asimmetrici), utilizza certificati e un meccanismo di MAC (Message Authentication Code).

Implementazione
Per prima cosa tramite l'SSL Handshake si crea un canale sicuro, affidabile e autenticato tra client e server. Successivamente l'SSL Record fa viaggiare i messaggi incapsulandoli in blocchi cifrati e autenticati. Il primo definisce il canale, ovvero una suite crittografico, che contiene: meccanismi di cifratura da usare in seguito, dati per la (mutua) autenticazione e relative chiavi. Il secondo implementa il canale utilizzando la suite per cifrare e autenticare i blocchi prima di darli in pasto a TCP.

Una sessione SSL è un'associazione tra un client e un server creata dal protocollo di handshake. Tale sessione è associata ad un insieme di parametri di sicurezza (come il session identifier, il peer certificate, il metodo di

compressione, il cipher spec e il master secret) utilizzati per evitare la costosa negoziazione di nuovi parametri di sicurezza.
Una connessione SSL è una connessione di trasporto tra un client e un server, tali connessioni sono di tipo transitorio ed ogni connessione è associata ad una sessione (lo stato di una connessione contiene delle sequenze random di bytes per identificare la connessione tra i due, una chiave segreta utilizzata per il MAC, la chiave di crittazione e quella di decrittazione, e altro). In generale tra un client-server vi è una singola sessione ma possibili multiple connessioni.

Protocollo di Handshake SSL
La Fase 1 prevede per prima cosa un "Client Hello": un client U manda al server S un messaggio richiedendo una connessione SSL, specificando le "prestazioni" di sicurezza desiderate, come ad esempio la versione del protocollo SSL supportato, la lista di algoritmi di compressione supportati, il cipher suite SSL_RSA_WITH_3DES_EDE_CBC_SHA (RSA: scambio chiavi di sessione; 3DES_EDE_CBC: cifratura simmetrica; SHA: funzione hash one-way per MAC). Infine il client allega una sequenza di byte casuali.
Sempre nella Fase 1 è previsto un "Server Hello" nel quale il server S riceve il messaggio "Client Hello" da U, seleziona quindi un algoritmo di compressione tra quelli elencati dal client e seleziona dalla cipher suite inviata da U una cipher suite comune (tra U e S). Il server S invia poi ad U un messaggio "Server Hello" contenente gli elementi selezionati e una nuova sequenza di byte casuali, se U non riceve il messaggio "server hello" interrompe la comunicazione.
Nota: durante la fase 1, il client e il server si scambiano bytes generati in modo casuale, sono utilizzati sia per costruire il *master secret* che per prevenire replay attacks.

La Fase 2 prevede l'autenticazione del server: il server S si autentica a U inviandogli il proprio certificato digitale, se i servizi offerti da S devono essere protetti negli accessi, S può richiedere a U di inviargli il suo certificato (autenticazione di U con S). L'SSL si basa su un PKI e non si interessa direttamente di meccanismi di trust, distribuzione dei certificati, revoca, etc. In fine il server S invia il messaggio "server hello done" a U e questo sancisce la fine della fase in cui ci si accorda sulla cipher suite e sui parametri crittografici.

La Fase 3 prevede la creazione del Pre-Master Secret e del Master Secret. Il client U costruisce un pre-master secret P come una nuova sequenza di byte casuali, spedisce poi P a S dopo averlo cifrato con la chiave pubblica di quest'ultimo contenuta nel certificato utilizzando l'algoritmo concordato (nell'esempio U usa RSA). Il client U combina P con alcune stringhe note più byte casuali contenuti in "client hello" e "server hello" e codifica il tutto con SHA e MD5 ottenendo il master secret M.
Il Server dovrà però anch'esso costruire il Master Secret: per prima cosa decifra il messaggio di U e ottiene P, calcola poi M nello stesso modo con cui U aveva

calcolato M a partire da P (può farlo perché dispone delle stesse informazioni di cui dispone U).
Successivamente è necessario che il client invii il proprio certificato ad S quando lo richiede, se non lo possiede, si interrompe il protocollo. Insieme al certificato, U allega e firma con la sua chiave privata, la SSL-history, il server S controlla il certificato e in caso di anomalie interrompe il protocollo.
L'autenticazione del client verso il server può avvenire o tramite certificato (come indicato sopra) o in caso U non ne abbia uno tramite login e password.

La Fase 4, l'ultima, tratta i Messaggi Finished: questi messaggi vengono costruiti in base al master secret e contengono tutte le informazioni che i due partner si sono
scambiati durante la fase di handshake. Permettono a U e S di effettuare un controllo ulteriore sulle comunicazioni avvenute e di accertarsi di possedere lo stesso
master secret, permettono inoltre ad U e S di accertarsi che non ci sia stato un attacco di tipo *man-in-the-middle*.
Il Client Finished: il client U invia ad S il messaggio "finished" protetto utilizzando M, la costruzione di "finished" avviene tramite: F_U = M + tutti i messaggi di handshake scambiati finora + identità di U, il client U codifica poi F_U con SHA e MD5 e lo invia ad S.
Il Server Finished: S verifica il messaggio finished di U ricalcolando il tutto ed invia ad U il suo messaggio "finished" protetto utilizzando M. La costruzione di "finished" avviene tramite: F_S = M + tutti i messaggi di handshake scambiati finora (incluso il messaggio "finished" di U) + identità di S. Il server S codifica F_S con SHA e MD5 e lo invia a U che verificherà il messaggio "finished" ricevuto da S.

Protocollo di Record SSL
Tale protocollo fornisce confidenzialità (il protocollo di handshake definisce una chiave segreta condivisa utilizzata per la criptazione dei payload SSL) e integrità (il protocollo di handshake definisce una chiave segreta condivisa utilizzata per generare il Message Authentication Code).

In questo protocollo i dati vengono frammentati in parti di lunghezza opportuna, ogni blocco viene poi numerato, compresso e autenticato tramite MAC. Successivamente cifrato con la chiave segreta e trasmesso utilizzando il protocollo di trasporto sottostante (TCP). Il destinatario opera in modo inverso al mittente e restituisce il messaggio all'applicazione sovrastante (HTTP, SMTP, IMAP, …). Il MAC nel protocollo Record SSL è nella forma MAC = H(blocco, numero, M, stringhe note), dove H è la funzione hash specificata, il numero è un identificativo di 64 bit che permette quindi non sia ripetuto all'interno di una sessione (2^{64} blocchi) prevedendo così l'uso fraudolento e iterato dello stesso blocco in una singola sessione.

3 - Autenticazione utente

User Authentication (Sicurezza nei Sistemi Operativi)

Quando avviene un primo contatto con un servizio (login, email, social network, …) è necessario identificarsi e poi autenticare questa identità per provare chi si dice di essere. L'autenticazione è la base per mettere in pratica l'autorizzazione.

L'autenticazione umana può avvenire tramite: something you know (password, PIN), something you have (security token), something you do (pattern), something you are (biometrics), where you are.

Autenticazione basata su password
Una password può sempre essere indovinata da un attacker. Sia P la probabilità di indovinare la password durante un intervallo di T unità di tempo. Sia G il tasso di tentativi per unità di tempo, ed N lo spazio delle password allora $P \geq (G * T) / N$.
Per abbassare P è possibile ridurre T (cambiare spesso pswd) o G, oppure aumentare N (pswd complesse).

Attacchi on-line: il sistema stesso è usato per verificare la correttezza dei tentativi.
Qui può essere ridotto G se il sistema impone un tasso di tentativi contenuto (es. 1 ogni 3 secondi), oppure limitando il numero di tentativi (ad es. massimo 4 tentativi). Attacchi off-line: si verifica la correttezza dei tentativi su un sistema diverso da quello targetizzato. Qui è necessaria una lista di possibili password costruita a priori (dictionary) ed è necessario avere accesso alle password memorizzate.
Invece che memorizzare la password cifrata con una chiave, è possibile memorizzare il digest risultante da una hash function one-way, e per l'autenticazione si fa il matching tra il digest salvato e quello prodotto dalla password immessa dall'utente durante il login.

Dictionary Attacks
Tramite una lista di parole comuni (dictionary) e una copia del file contenente le password criptate (digests), per ogni parola w nel dizionario si calcola il suo digest e lo si compara al digest della password da trovare. Questo si può rendere più sofisticato utilizzando combinazioni di parole conosciute (invece che provare con parole generate da lettere a caso). Può essere meccanizzato con programmi come *crack*.

Le difese possono essere limitare l'accesso alle password nel SO, rallentare artificialmente le performance della funzione one-way hash che è utilizzata (Unix utilizza 25 volte il DES ad un blocco di tutti zeri con la password come

chiave), utilizzare il “salting” delle password per prevenire attacchi globali, applicare il “shadow” alle passwords ovvero separare le password criptate (in un file leggibile solo a root) dalle altre informazioni tipicamente contenute nel file di password (es. vero nome dell’utente, indirizzo, numero di telefono,...) leggibile a tutti.

Salting
Quando l’utente U fornisce per la prima volta la password P, il sistema associa all’utente U due sequenze S e Q. S, detto seme (salt), è un numero generato a caso dal sistema; Q è il risultato di f(P | S) dove f è una funziona hash one-way. S e Q (non P) vengono memorizzate associate all’utente U.
Quando l’utente U vuole collegarsi al sistema e fornisce la sua password P, vengono recuperati per prima cosa S e Q associate all’utente U, si concatena S con P e si applica f ottenendo Q*. Si confronta poi Q* con Q e se Q* = Q allora il collegamento riuscito, altrimenti no. Se qualcuno accede al file delle password, ottiene S e Q ma
dai quali non riesce a risalire a P. Tips: in Unix il salt (lungo 12 bits) è utilizzato per modificare leggermente la funzione E-Box interna al DES, ed è memorizzato come stringa di 2 caratteri nel file delle password. Utilizzando salting diversi per ogni utente impedisce di capire se due utenti hanno utilizzato la stessa password.

Shadow Passwords
Il file /etc/passwd è leggibile a tutti perché contiene informazioni che vanno al di là della password, il meccanismo delle shadow password infatti memorizza le password in un file separato /etc/shadow, leggibile solo a root.

Login Spoofing
L'attaccante scrive un programma (testuale o grafico) che presenta una finta schermata di login, attende che la vittima inserisca login/password, poi memorizza o si spedisce la coppia login/password, visualizza un messaggio di “login incorrect” e fa partire il vero programma di login, per esempio terminando la shell attuale.
La vittima crede di aver digitato male la password, questa volta entrando senza problemi nel sistema.
Alcune *difese* generali contro il login spoofing sono basate sulla mutua autenticazione: l’utente si autentica sull’host e l’host si autentica con l’utente (sono coinvolte tecniche crittografiche come firme digitali e certificati).

Phishing
È una “moderna” incarnazione del login spoofing, i phishers tentano di acquisire in modo fraudolento informazioni sensibili come passwords e carte di credito mascherandosi come una persona/business affidabile. Tipicamente si manifesta

tramite email o instant messaging (social engineering). Lo spear-phishing (arpionamento) è un attacco di phishing mirato ad individui specifici.

Keyloggers
I keyloggers sono designati come spyware e si manifestano nella forma di Cavalli di Troia, possono registrare password, riconoscono quando si digitano caratteri. Sono anche utilizzati per tracciare le abitudini di navigazione e possono esistere sia versioni software che hardware.
Alcune difese possono essere programmi di spyware-detection o firewall che bloccano il traffico di rete in uscita o anche tastiere virtuali.

Packet Sniffing
Un packet sniffer è un software che analizza il traffico di rete su cui l'host è collegato.
Cerca di individuare pacchetti contenenti coppie login/password spediti "in chiaro" da meccanismi di comunicazione come telnet, rlogin e ftp. Memorizza le coppie login/password per un uso futuro.
Le pratiche generali di difesa sono basate su tecniche crittografiche di oscuramento delle password o one-time passwords.

Tecniche Crittografiche di Autenticazione

Challenge-Response (schema generale)
L'utente U dichiara la propria intenzione di accedere all'host, il quale seleziona una "challenge" e la ritornerà ad U. Questo calcola la "risposta" alla challenge e la ritorna all'host, che comparando la risposta ricevuta da U con la risposta "attesa" per la challenge che ha inviato fornirà o meno l'accesso in caso le due combacino. Questo è uno schema OTP (one-time-password), dal momento che la risposta è unica per la challenge e può essere utilizzata una sola volta (in quanto la challenge cambia ogni volta).

Challenge-Response (con crittografia simmetrica)
L'utente U e l'host condividono una chiave segreta K (password), U dichiara la propria intenzione di accedere all'host, il quale genera e memorizza una stringa random *chal* e la invia ad U. Quest'ultimo calcola $resp = C_K(chal)$ [*chal* criptata con la chiave K] e la invia all'host come risposta alla challenge, l'host compara *chal* con $D_K(resp)$ [*chal* decriptata con la chiave k] e se combaciano, l'accesso è fornito, altrimenti no. Dato che solo U (e l'host) conoscono K, l'autenticazione è assicurata.

Challenge-Response (con crittografia asimmetrica)
L'host mantiene un file di ogni chiave pubblica degli utenti, quando l'utente U dichiara la propria intenzione di accedere all'host, questo genera una stringa random *chal* e la invia ad U che firma la challenge e la ritorna all'host come

risposta *resp = Sign(chal)*. L'host verifica poi la firma *Verify(resp)* e se è valida, l'accesso è fornito, altrimenti no (le proprietà delle firme digitali assicurano l'autenticazione).
Tips: gli algoritmi di challenge-response sono simili alle tecniche IFF (identification friend or foe) che gli aerei militari utilizzano per identificare alleati o nemici.

Pass Algorithms
Assumendo un sistema di autenticazione challenge-response nel quale la funzione f è segreta, allora f è chiamata pass algorithm. Secondo questa definizione, non è richiesta nessuna chiave crittografica o informazione segreta come input di f. È l'algoritmo stesso che calcola f segreto.

One-Time-Password (con funzioni "One-Way Hash")
Una one-time-password è una password che viene invalidata subito dopo l'utilizzo. Anche questo è un meccanismo challenge-response, la challenge è il numero del tentativo di autenticazione, il response è la one-time-password. L'host genera un numero random R per l'utente U e una stringa di numeri partendo da R: $x_0 = R$, $x_1 = f(x_0)$, $x_2 = f(x_1)$, $x_3 = f(x_2)$,... dove f è una funzione hash. U porta con se (materialmente) x_0,...,x_{99} e l'host memorizza (in chiaro) x_{100}. A questo punto per accedere all'host, U invia il proprio nome e x_{99} (in chiaro), l'host riceve (U, y) e calcola f(y) e lo compara con il valore memorizzato per l'utente U (che è x_{100}). Se i due valori combaciano, l'accesso è fornito (l'host deve aver ricevuto x_{99}), altrimenti no. Infine U cancella x_{99} dalla propria lista e l'host rimpiazza x_{100} con x_{99}.
Tips: I token fisici calcolano il "response" con f(t|k) dove *f* è una funzione one-way hash, *t* è il tempo attuale (in minuti) e *k* è una chiave (segreta) inserita nel token (che è associata all'account utente quando la banca fornisce il token).

Strong Authentication
Combina due elementi tra: something you know, something you have e something you are (la combinazione tra i primi due è la più usata, ad es. password+token)

2-Step Verification
One-time-password (es. codice SMS) usato in combinazione con login/password crea una forma di strong authentication conosciuta come 2-step verification (o 2-factor authentication).

4 - Controllo d'accesso a oggetti

Controllo degli Accessi (Sicurezza nei Sistemi Operativi)

L'autorizzazione è l'insieme delle politiche e dei meccanismi utilizzati per valutare se un particolare soggetto è autorizzato a compiere particolari operazioni su un oggetto.
Principio del Least Privilege: ogni utente o programma dovrebbe operare utilizzando il minimo insieme di privilegi necessari per fare il proprio lavoro. Tale principio è imposto attraverso un reference monitor che implementa completa mediazione, ogni accesso ad ogni oggetto è controllato.

Trusted Computing Base (TCB)
Il TCB è ogni cosa in un sistema di computing che fornisce un ambiente sicuro. Questo include il sistema operativo e i meccanismi di sicurezza che fornisce, hardware, locazioni fisiche, hardware di rete e software, e procedure prescritte. Tipicamente ci sono disposizioni con il compito di controllare l'accesso, fornire autenticazione a risorse specifiche, supportare l'autenticazione utente, proteggere da virus e altre forme di infiltrazione nel sistema, e backup dei dati.
È presente l'insieme dei soggetti S (le entità attive che eseguono azioni), l'insieme degli oggetti O (le entità passive/attive su cui vengono eseguite azioni; $S \subseteq O$) e l'insieme delle modalità di accesso (le operazioni che possono essere eseguite).

Alcuni esempi, nei sistemi operativi i soggetti sono processi e thread, gli oggetti sono le risorse di sistema (file, dispositivi, processi) e le modalità di accesso: read, write, execute. Oppure nei sistemi object-oriented i soggetti sono i processi e i thread, gli oggetti sono appunto gli oggetti e le classi nel senso OO e le modalità di accesso sono i metodi definiti nelle classi.

Principi fondamentali
- Principio di "Fail-safe defaults": nessun soggetto ha diritti per default.
- Principio di "Privilegio minimo": ogni soggetto ha, in ogni istante, i soli diritti necessari per quella fase dell'elaborazione
- Principio di "Accesso Mediato" (reference monitor): tutti gli accessi ad un oggetto devono essere controllati.

Un *dominio di protezione* è un insieme di oggetti e i tipi di operazioni che possono essere effettuati su ogni oggetto.
Formalmente, è un insieme di coppie del tipo <oggetto, insieme di operazioni>, ogni soggetto opera all'interno di un dominio di protezione.

Le modalità "kernel mode" e "user mode" possono essere viste come due domini di protezione di accesso alla memoria centrale. Normalmente i processi operano

in user mode, quando attivano una system call, attuano un cambio di dominio a kernel e cambiano i diritti di accesso. Questo è un esempio di associazione dinamica.

Matrice di accesso
È una matrice di domini/oggetti in cui l'elemento access(i, j) rappresenta l'insieme dei diritti che il dominio D_i prevede per l'oggetto O_j.
Quando viene creato un nuovo oggetto si aggiunge una colonna alla matrice di accesso e il contenuto di questa colonna è deciso al momento della creazione dell'oggetto.

Implementazione: tabella globale
Per quanto riguardo il metodo della tabella globale consiste nell'effettiva memorizzazione di una tabella di terne <dominio, oggetto, insieme dei diritti>. I vantaggi sono che è di semplice realizzazione mentre gli svantaggi che la tabella diventa molto grande e risulta difficile l'aggiornamento dei permessi di accesso (e.g. togliere a tutti il permesso x a un oggetto y).

Implementazione: access control list (ACL)
Con il metodo ACL viene memorizzata per colonne, ad ogni oggetto viene associata una lista di elementi <dominio, diritti di accesso>. Alcune ottimizzazioni possibili sono la riduzione dell'ampiezza della lista associando i diritti a insiemi di domini o usando diritti standard (di default). Ad esempio UNIX ha liste di soli 3 elementi: owner, group, others.

Implementazione: matrice di accesso
La tabella viene memorizzata per righe, ad ogni dominio viene associata una lista di elementi <oggetti, diritti su tali oggetti>; queste coppie sono dette *capability*.
Le capabilities sono mantenute dai processi che le presentano come "credenziali" per accedere all'oggetto, sono una sorta di "chiavi" per l'accesso alla "serratura" che protegge l'oggetto.
Perché il meccanismo delle capability funzioni occorre che i processi non possano "coniare" ad arte capability false, l'oggetto (reference monitor) possa riconoscere le capability autentiche e sia possibile negare a un processo il diritto di copia o
cessione della propria capability ad un altro processo.

Capability
Possono essere implementate tramite tecniche crittografiche: al processo viene fornita come capability la tripla <oggetto, diritti di accesso, codice di controllo> cifrata con una chiave conosciuta solo all'oggetto. Il processo può memorizzare la capability ma non può modificarla (è per esso una stringa di bit indecifrabile).
Quando un processo vuole accedere a una risorsa presenta la richiesta all'oggetto insieme con la capability relativa. L'oggetto decodifica la capability,

verifica che il codice di controllo sia corretto e che la richiesta del processo sia autorizzata dalla capability ricevuta. La capability può anche essere trasferita ad un altro processo. I meccanismi utilizzati per proteggere le capabilities sono tags, memoria protetta e crittografia.

Revoca
La revoca può essere: immediata o ritardata (subito o si può attendere), selettiva o generale (per alcuni i domini o per tutti), parziale o totale (tutti i diritti o solo alcuni) e temporanea o permanente.
Per la revoca in sistemi basati su ACL è sufficiente aggiornare in modo corrispondente le strutture dati dei diritti di accesso.
Per la revoca in sistemi basati su capability l'informazione relativa ai permessi è memorizzata presso i processi. In questo caso la capability ha validità temporale limitata: una capability scade dopo un prefissato periodo di tempo è quindi così permesso revocare diritti (ma in modo ritardato).

Le capability indirette: vengono concessi diritti non agli oggetti ma a elementi di una tabella globale che puntano agli oggetti ed è possibile revocare diritti cancellando elementi della tabella intermedia.

Controllo degli accessi in UNIX
Ogni oggetto (risorsa) in UNIX è un file con uno schema nominativo strutturato ad albero, ad ogni file è associato: un utente proprietario del file (owner) e un insieme di utenti (group).
Gli utenti e i gruppi sono identificati con interi letti dal password file, presenti nella forma "user-id:group-id".

Proprietà di file e processi
Ogni processo creato dall'utente (programma in esecuzione) passa l'user-id e il group-id al processo. Quando un processo crea un nuovo file, gli viene assegnato l'id dell'utente e del gruppo dell'utente che ha avviato il processo. Per cambiare proprietario di un file si applica: chown newusername file(s).

User ID Reale vs Effettivo
Ogni processo possiede diverse ID associate ad esso:
- real-user-id, real-group-id (identificano il vero utente e gruppo che esegue il processo, questi valori sono presi dalla entry nel passwd file e non cambiano durante la vita di un processo).
- effective-user-id, effective-group-id (sono quelle effettivamente utilizzate per determinare i diritti di accesso al file system, sono stabiliti dinamicamente nel corso dell'esecuzione del processo tramite il meccanismo di setuid)

Oltre a real-user-id, real-group-id, effective-user-id e effective-group-id, ogni processo possiede un saved-user-id e saved-group-id che contiene copie

dell'effettivo user-id ed effective-group-id che esisteva al momento in cui un programma setuid è stato eseguito. Saved-user-id e saved-group-id permettono al processo di ritornare al proprio effettivo user/group-id una volta che l'esecuzione del programma setuid termina.

Di norma effective-user-id e real-user-id sono uguali, così come effective-group-id e real-group-id. Al momento in cui viene eseguito un file eseguibile con il bit set-user-id del suo insieme di permessi settato, saved-user-id viene impostato all'effective-user-id, mentre effective-user-id è impostato all'user id del proprietario del file. Al momento in cui viene eseguito un file eseguibile con il bit set-group-id del suo insieme di permessi settato, saved-group-id viene impostato all'effective-group-id, mentre effective-group-id è impostato al group id del proprietario del file.

Questi meccanismi permettono agli utenti di eseguire un eseguibile con i permessi del proprietario/gruppo dell'eseguibile, i nuovi permessi rimangono effettivi durante il corso dell'esecuzione permettendo ad un processo di modificare il proprio dominio di protezione dinamicamente durante l'esecuzione. Questo può essere utilizzato per implementare il principio del last privilege.

setuid è un esempio di implementazione del comando /bin/passwd che permette agli utenti di cambiare le proprie password. Un utente dovrebbe essere in grado di cambiare la propria password ma non dovrebbe essere in grado vedere (o modificare) le password di altri. In Unix, i permessi sono alla granularità di un intero file, è comunque possibile definire permessi alla granularità di records individuali (ad esempio le linee nel file /bin/passwd). Utilizzo di setuid per risolvere il problema della password: il comando /bin/passwd è di proprietà di root con i permessi r-s--x--x (il bit setuid è abilitato), invece il file /etc/shadow è, si, di proprietà di root ma con i permessi rw------- (read/write root). Quando /bin/passwd viene eseguito da un processo utente, il suo effective-user-id diventa quello di root e quindi il comando può scrivere su /etc/shadow ma solo dopo aver fatto tutti i controlli necessari implementati dal programma stesso.

5 - Esostrutture

Firewalls

Combinazione di hardware e software per regolare il traffico tra una rete interna e una esterna (es. internet). Un firewall si colloca ogni volta che si hanno due reti separate comunicanti che si vogliono tenere controllate.

Un firewall può forzare politiche di sicurezza, concentrarsi su cosa si vuole far entrare e uscire dalla rete, registrare le attività su internet (log). Mentre non può

proteggere da qualcuno che è già entrato nella rete, proteggere da connessioni che lo bypassano, proteggere da minacce nuove (sconosciute), proteggere da virus e worms, settarsi da solo correttamente.

Tecnologie Firewalls: Packet filtering
Implementato attraverso uno screening router (packet filter): mentre un router risponde alla domanda "il pacchetto può essere trasportato fino alla destinazione?", uno screening router risponde alla domanda "dovrebbe il pacchetto essere trasportato fino a destinazione?".
In sostanza applica un insieme di regole ad ogni pacchetto in arrivo/uscita e poi lo inoltra o lo scarta.

Tecnologie Firewalls: Stateful Packet Filtering
Oltre ad osservare l'intestazione dei pacchetti possono anche mantenere uno stato in memoria di ogni pacchetto. Informazioni come se un pacchetto è una risposta ad un pacchetto arrivato in passato, numero di pacchetti arrivati dallo stesso host, se il pacchetto è un frammento, se è identico ad uno già arrivato, ed altro.
I vantaggi sono che uno *screening router* può proteggere un intero network, è estremamente efficiente e disponibile ovunque. Gli svantaggi sono che è difficile da configurare, riduce le performance in termini di velocità e le politiche che possono essere settate sono molto limitanti.

Tecnologie Firewalls: Proxy Servers
Programmi applicativi specializzati per servizi internet (HTTP, FTP, telnet,…) che vengono forniti da un server "di mezzo" con lo scopo di restringere le comunicazioni dirette tra reti interne ed esterne. Invece che accedere direttamente al server, si accede ad un altro, un proxy, che tramite caching risponde alle richieste.
Il proxy può essere anche utilizzato per "controllare" il traffico, in modo che determini per ogni accesso se rispondere alla richiesta o meno.
I vantaggi sono l'autenticazione a livello-utente, funge da filtro intelligente, può essere combinato con il caching e può effettuare un buon logging. Gli svantaggi sono che richiede diversi servers per ogni servizio e necessita di modifiche ai clients.

Tecnologie Firewalls: Network Address Translation (NAT)
Permette a una rete di utilizzare un insieme di indirizzi interni e un diverso insieme di indirizzi esterni. Tale metodo è stato inventato non per sicurezza ma per conservare indirizzi IP e viene tipicamente implementato all'interno di un router.
I vantaggi sono che restringe il traffico in entrata, nasconde la struttura e i dettagli di una rete interna e forza il controllo del firewall sul traffico in uscita.
Gli svantaggi sono che interferisce con alcune tecniche basate sulla crittografia,

l'allocazione dinamica degli indirizzi interferisce con il logging e la rete interna non può hostare servizi visibili esternamente (servirebbe port mapping). Il Network Address Translation con port forwarding permette di creare servizi all'interno di una rete NAT visibili dall'esterno.

Architetture Firewall: Host-based Firewalls
È un firewall applicato ad un singolo host, viene comunemente utilizzato per proteggere i servers. I vantaggi sono l'alta personalizzabilità, è indipendente dalla topologia (gli attacchi interni ed esterni devono attraversare il firewall) ed è estensibile, nuovi servers possono essere aggiunti alla rete con il proprio firewall, senza alterare la configurazione di rete.

Architetture Firewall: Personal Firewalls
È un programma installato su un comune personal computer (PC) che controlla le comunicazioni in entrata e in uscita dal PC stesso, permettendo o vietando alcuni tipi di comunicazione in base a regole o policy di sicurezza preimpostate dall'utente in fase di configurazione.

Architetture Firewall: Screening Router
Lo screening router firewall è anche conosciuto come firewall di livello rete o packet-filter. Questo firewall seleziona (screening) i pacchetti in entrata attraverso attributi del protocollo. Gli attributi del protocollo selezionati possono includere indirizzi di origine o di destinazione, il tipo di protocollo, la porta di origine o di destinazione o altri attributi specifici del protocollo.

Architetture Firewall: Dual-Homed Host
Questo firewall fornisce la difesa di prima linea e tecnologie di protezione per impedire a corpi estranei di compromettere la sicurezza delle informazioni violando uno spazio di rete fidato. Un dual-homed host è un sistema costituito da due interfacce di rete (NICs) che risiede tra una rete inaffidabile (es. internet) e una rete fidata (es. rete aziendale) con lo scopo di fornire un accesso sicuro. Dual-homed è un termine generico per proxies, gateways, firewalls, o qualsiasi altro server che fornisce applicazioni o servizi sicuri direttamente a una rete non fidata.
Può essere visto come caso speciale di Bastion Host e Multi-Homed Host.

Architetture Firewall: Screened Host
L'architettura screened host viene realizzata mediante più componenti fisici. L'elemento principale dal punto di vista della protezione della rete privata è uno screening router, i servizi vengono forniti da un calcolatore (il bastion host) appartenente alla rete interna che svolge funzioni di application gateway. In questa architettura il bastion host ha una sola interfaccia di rete e la separazione della rete interna viene realizzata dal router, che filtra i pacchetti in maniera tale che solo il bastion host possa aprire connessioni con la rete esterna. Viceversa,

tutti i sistemi esterni che desiderino collegarsi con la rete privata possono connettersi solo con il bastion host. Questa architettura è molto più elastica della precedente e sono possibili due approcci: o impedire tutti i tipi di connessione da e verso host interni, eccetto il bastion host, e quindi forzare l'utilizzo dei proxy server sul bastion host (mediante un opportuno filtraggio da parte del router) o permettere ad altri host interni di aprire connessioni con Internet per alcuni specifici servizi. Questi due approcci possono essere combinati in modo da realizzare una politica di sicurezza della rete che si avvicini quanto più possibile a quella desiderata.

Architetture Firewall: Screened Subnet
Una screened subnet (aka "triple-homed firewall") è un'architettura che utilizza un singolo firewall con tre interfacce di rete. L'interfaccia 1 è l'interfaccia pubblica e si connette ad internet. L'interfaccia 2 si connette ad una DMZ (DeMilitarized Zone) nella quale i servizi pubblici hostati sono collegati. L'interfaccia 3 si connette ad un intranet per accedere da e verso reti interne. Anche se il firewall stesso fosse compromesso, l'accesso all'intranet non dovrebbe essere disponibile, purchè il firewall sia configurato correttamente.

Nota: un bastion host è un computer in rete specificamente progettato e configurato per resistere ad attacchi, in quanto vi è largamente esposto. Tutto il traffico attraversa il bastion host che può controllarlo e bloccarlo/inoltrarlo. Generalmente vengono eseguite poche applicazioni, principalmente proxies.

Tips: la DMZ (De-Militarized Zone) è una porzione di rete che separa una rete puramente interna da una esterna. È qui che può essere inserito eventualmente un honeypot.

6 - IPSec

IPSec applica una sicurezza al livello più basso possibile (livello IP), tramite l'inserimento di uno strato di IP con alcune caratteristiche di sicurezza, invece dell'IP classico. Questo però richiede certe modifiche al SO.

IPSec è una famiglia di protocolli, creati dal Internet Engineering Task Force (IETF), creati per tappare falle di sicurezza nell'IP creato alla nascita di internet. A quest'ultimo mancava integrità, autenticazione e confidenzialità (violabili tramite IPSpoofing e Packet Sniffing).

I protocolli IPSec sono basati su *extension headers* (i protocolli di base sono AH, ESP e IKE).
Le applicazioni principali sono Virtual Private Networks (VPN) su internet (simulazione di una rete privata all'interno di una rete pubblica come internet), accesso da remoto ad una rete attraverso internet, stabilire una connettività

extranet e intranet con i partners (altre organizzazioni ad esempio, assicurando autenticazione e confidenzialità, fornendo un meccanismo di scambio di chiavi).

I benefici sono che quando IPSec è implementato su un firewall o un router, fornisce una forte sicurezza a tutto il traffico che attraversa il perimetro (nessun impatto però sul traffico interno). Inoltre non necessita modifiche a livello software nel sistema di un utente o un server.

I protocolli sono:
- Authentication Header (AH) - gestisce autenticazione ed integrità
- Encapsulating Security Payload (ESP) - per la confidenzialità
- Internet Security and Key Management Protocol (IKE) - per lo scambio di chiavi

I primi due sono applicati "per pacchetto" ed entrambi supportano transport mode e tunnel mode.

Il Transport Mode fornisce protezione ai protocolli del livello soprastante (payloads dei pacchetti IP) ed è di solito utilizzato per le comunicazioni end-to-end. AH in transport mode autentica l'IP payload e porzioni selezionate dell'IP header. ESP in transport mode cripta e opzionalmente autentica l'IP payload, l'IP header non è protetto. Questa modalità ha un basso overhead ma alcune informazioni potrebbero essere sniffate

Il Tunnel Mode fornisce protezione all'intero pacchetto IP ed è di solito utilizzato per le comunicazioni gateway-to-gateway. Gli headers AH/ESP sono aggiunti al pacchetto IP e l'intero pacchetto è trattato come il payload di un nuovo pacchetto IP con un nuovo outer IP header, il tutto è indirizzato verso un gateway. Questa modalità è più sicura, ma necessita di entità intermedie ed ha un overhead più alto.

Esempio di tunneling:
L'host A è su una rete N_A (aziendale) e vuole comunicare con l'host B sulla rete N_B. A genera un pacchetto con A come mittente e B come destinazione, il pacchetto è instradato verso il security gateway (firewall con IPSec) della rete N_A. Quest'ultimo incapsula il pacchetto in un outer IP header con N_A come mittente e N_B come destinatario. Il nuovo pacchetto è instradato dalla rete pubblica (internet) al security gateway del network N_B, il quale estrae, decripta e autentica il pacchetto originale che sarà poi instradato e consegnato a B su N_B.

AH: nell'autenticazione IPSec le due parti devono condividere una chiave segreta, AH fornisce un supporto obbligatorio al DES (ma anche Triple-DES, RC5, IDEA, CAST, Blowfish).
ESP: così come AH anch'esso supporta l'utilizzo del MAC, utilizzando l'HMAC.

7 - Denial of Service

Con disponibilità ci si riferisce alla possibilità di utilizzare una risorsa o un servizio desiderato. Un attacco Denial of Service tenta di rendere l'informazione/risorsa/servizio richiesta/o inaccessibile agli utenti.

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)
È un tipo di test challenge-response utilizzato per determinare se l'utente è umano.
CAPTCHA implica che un computer (un server) chieda ad un utente di completare un test, il quale può essere generato e validato da un computer ma un computer non è in grado di risolverlo.

reCAPTCHA
Ha lo scopo di riutilizzare lo sforzo umano impiegato per risolvere i CAPTCHA, ad esempio mettendo come immagini da risolvere foto di libri scannerizzati male che un computer non sarebbe in grado di capire, o numeri civici non ben leggibili.

noCAPTCHA
È possibile distinguere umani da computer in base a come muovono il puntatore prima e dopo aver cliccato una certa textbox.

Tipi di DoS
Le due strategie generali per gli attacchi sono il *crash* o il *flood* dei servizi. I metodi più utilizzati per lanciare un attacco sono consumare tutta la larghezza di banda disponibile, consumare le risorse dell'host (RAM, spazio su disco, tempo CPU), disturbare le informazioni di configurazione (es. routing), disturbare le informazioni di stato (es. sessioni TCP), disturbare i componenti di una rete fisica (LAN, WLAN, …).
Tips: oggigiorno gli attacchi DDoS sono per lo più effettuati tramite botnets di zombies (insieme di computer compromessi che operano insieme per effettuare un attacco congiunto). Inoltre tali attacchi fanno affidamento su indirizzi IP spoofati e IP headers corrotti.

Attacchi conosciuti
- *Ping of Death*: l'attacker crea pacchetti IP maggiori di 64KB (65536 bytes), il limite definito dal protocollo IP, sfruttando dei bug di alcune precedenti implementazioni di TCP/IP può creare un ping malformato. Tali attacchi sono risolti grazie a nuove patch dei sistemi operativi e firewalls.
- *Teardrop*: si sfrutta la frammentazione dei pacchetti IP, ogni pacchetto frammentato possiede un offset che permette all'intero pacchetto di essere riassemblato, l'attacker invia frammenti IP malformati con overlapping

(sovrapponimento) e payloads over-sized alla macchina obiettivo dell'attacco, causando un crash.
- *SYN Flooding*: sfrutta le vulnerabilità dell'handshake a tre-vie TCP attraverso IP spoofing, l'attacker, attraverso la botnet, inizializza molte richieste di connessione TCP inviando SYNs all'host vittima. La vittima inizializza le connessioni nel
Transmission Control Block (TCB), invia i SYN-ACKs e attende gli ACK prima che ogni connessione venga dichiarata come "stabilita". Dal momento che la connessione iniziale è stata spoofata, i messaggi SYN-ACK sono persi e gli ACK non arriveranno mai. La serie ci connessioni in entrata nel TCB verrà riempita e nessun'altra connessione da altri host sarà accettata (attacco ottobre '16).
- *Reflector attack*: è una variazione dell'attacco SYN Flood utilizzando l'handshake TCP a tre-vie con IP spoofing, l'attacker, attraverso la botnet, inizializza varie richieste di connessioni TCP con molti host (reflectors) dove però l'indirizzo sorgente è spoofato con quello della vittima. Ogni reflector invia il proprio messaggio SYN-ACK alla vittima (spoofata), inondandola.
- *Smurf*: sfrutta le vulnerabilità dell'Internet Control Message Protocol (ICMP), l'IP Spoofing ed alcuni errori nelle configuazioni del broadcast di rete. L'attacker invia molti pacchetti di echo-richieste ICMP all'indirizzo di broadcast di una subnet (utilizzato per motivi di diagnostica della rete). Questi pacchetti contengono indirizzi IP spoofati all'indirizzo della vittima e sono inoltrati a tutti gli host della subnet, ogni host risponde inviando un flusso di pacchetti echo-reply alla vittima, intasandola.
- *Slow HTTP DoS*: sfrutta una vulnerabilità dei web servers thread-based (come Apache) che attendono che gli tutti gli headers HTTP siano ricevuti prima di rilasciare la connessione. Mentre i server solitamente utilizzano timeouts per terminare richieste HTTP incomplete, il timeout, settato solitamente a 300 secondi, è resettato appena il client invia dati aggiuntivi, mantenendo la richiesta HTTP aperta e inviando dati aggiuntivi prima che il timeout sia raggiunto, la connessione HTTP rimane aperta. Se un attacker riesce ad occupare le connessioni HTTP di un web server, gli utenti legittimi non sono in grado di accedere ai servizi forniti.

Difese
Le principali tecniche di difesa da attacchi DoS sono Attack Prevention (prima dell'attacco), Attack Detection and Filtering (durante l'attacco), Attack Source Traceback and Identification (durante e dopo l'attacco).
La *prevenzione* riduce la possibilità di diventare zombie, installando patches di sicurezza, antivirus e IDSs, mantenendo aggiornati protocolli e sistemi operativi, firewalls, ecc.
La *rilevazione* (detection) può avvenire attraverso l'identificazione di pattern statistici degli attacchi DDoS e la loro comparazione con il traffico attuale. L'identificazione di deviazioni dal comportamento standard dei clients e il

normale traffico di rete (detection anomaly-based), oppure un approccio ibrido tra le due.
Il *filtraggio* è più efficiente, tanto più è vicino all'attacker (isolando gli zombie, è tuttavia impossibile). Si può effettuare un filtro preventivo identificando gli IP spoofati (l'indirizzo IP sorgente del traffico in uscita dovrebbe appartenere alla subnetwork di origine, quello in entrata invece no).

Intrusion Detection

Un'intrusione può essere definita solo avendo già dichiarato politiche di sicurezza da mettere in atto, in caso tali politiche vengano violate allora si può parlare di intrusione.
Un Intrusion Detection System è un software, un elemento hardware o entrambi aggiunti al sistema con lo scopo di rilevare e avvertire in caso di intrusioni, idealmente, quando avvengono e rispondere prontamente.

Un falso positivo si manifesta quando avviene qualcosa di anormale secondo l'IDS, ma non è un intrusione.
Un falso positivo si manifesta quando un intrusione è in atto ma l'IDS non la rileva.
In generale l'IDS ha il compito di minimizzare falsi positivi e negativi, anche se spesso vi è una relazione inversa tra i due (al diminuire degli uni aumenta il numero degli altri). Un buon IDS dev'essere indipendente dalla supervisione umana, il suo funzionamento trasparente, dovrebbe applicare un overhead minimo al sistema,…

Modello di intrusione: misuse detection
L'IDS controlla le attività e in caso di comportamenti simili a tecniche di intrusione note (firme) o vulnerabilità del sistema, dà l'allarme. Ha generalmente pochi falsi positivi, è abbastanza veloce ed affidabile. Non è però in grado di proteggere da nuovi attacchi.

Modello di intrusione: anomaly detection
L'IDS controlla le attività e dà l'allarme in caso di comportamenti diversi da ciò che si aspetta un utente normalmente faccia. È più flessibile, può migliorare le performance, ma risulta difficile identificare le metriche da includere nel modello base (spesso necessita di "allenamento").

Note: altri modelli di intrusion detection sono reti neutrali, tecniche di classificazione machine learning, sistemi che imitano sistemi immunitari biologici.

Caratterizzazione IDS: host based
L'IDS controlla i dati da un singolo host utilizzato per rilevare le intrusioni. Di norma monitora il sistema, gli eventi e i security logs; è in ascolto in alcune porte e avverte quando sono utilizzate, controlla le firme tramite potenti reg-ex. Può rilevare attacchi come brute-force login e keyboard attacks, ha tuttavia un alto costo in quanto va installato su ogni host che si vuole proteggere

Caratterizzazione IDS: multi-host based
L'IDS controlla i dati da vari host, utilizzati per rilevare le intrusioni. I controlli sono simili all'host based.

Caratterizzazione IDS: network based
L'IDS controlla i dati del traffico di rete provenienti da vari host. Utilizza i pacchetti di rete come dati da analizzare e usa patterns e anomalie statistiche per rilevare gli attacchi. Ha un basso costo e può rilevare attacchi come DoS basati su IP e Payload Content, è indipendente dai SO ed effettua un controllo real-time. Tali IDS sono posizionati fuori dal DMZ.

Honeypots
I sistemi honeypot sono dei server esca o sistemi (da fuori apparentemente comuni) impostati per fornire informazioni riguardo un attacker o un intruso nel sistema.
È necessario che un honeypot risulti completamente irriconoscibile dall'attacker. Evitare che l'honeypot sia sfruttato dall'attacker come mezzo per altri attacchi (blocco traffico in uscita). Di norma un honeypot è messo nella DMZ.

Cyber Forensics

I passi da seguire sono:
- analisi di tutte le informazioni disponibili (quali attacchi sono stati usati, come ha fatto l'attacker ad ottenere l'accesso, come si comporta ora l'attacker)
- comunicazione con le parti pertinenti (sistemi utilizzati dall'attacker, sistemi sfruttati dall'attacker per confondere le proprie tracce, denuncia alla polizia e CERT nazionale)
- raccolta e protezione delle informazioni (salvaguardia della catena di conservazione delle prove)
- contenimento dell'intrusione (arresto temporaneo del sistema, disattivazione dell'accesso, verifica che i sistemi di backup non siano stati compromessi)
- eliminazione dei metodi di accesso degli attaccanti (cambio delle password, re-installazione sistemi compromessi, eliminazione di eventuali backdoor, virus, ecc)
- riportare il sistema al normale funzionamento

8 - Sicurezza nelle reti wireless

Le reti wireless presentano un numero di rischi sempre maggiore per varie cause, tra cui utilizzo di broadcasting nelle comunicazioni, che è molto più suscettibile a intercettazioni e interferenze, sono più vulnerabili agli attacchi attivi, i dispositivi wireless sono spesso mobile con risorse limitate.

Le *minacce* più diffuse sono:
- Associazione accidentale: connessione automatica a WLANs vicine
- Associazione malevola: access points finti
- Reti ad hoc: reti p2p tra i dispositivi
- Reti non tradizionali: rischio di MAC spoofing
- Attacchi man-in-the-middle: tra l'utente e l'access point
- Denial of Service: è particolarmente semplice indirizzare messaggi wireless ad un target.
- Network injection: access points che sono esposti a protocolli di routing o messaggi di gestione della rete
- Sniffing: particolarmente facile intercettare pacchetti

IEEE 802.11
È un comitato che ha sviluppato degli standard per un'ampia gamma di LANs. Disponibile nelle versioni 802.11b, 802.11g, 802.11ac, 802.11n.
Alcune tecnologie per aumentare la sicurezza di una rete wireless sono WEP e WAP basate sullo stream cipher RC4. Successivamente è stato sviluppato WPA2, più sicuro dei precedenti (aka Robust Security Network RSN, basato sul block cipher AES).

Port-Based Network Access Control (PNAC): fornisce un meccanismo di autenticazione ai dispositivi che desiderano allacciarsi ad una LAN (ha lo scopo di limitare l'accesso alla rete). Su questo meccanismo vi è lo standard IEEE 802.1X basato sull'Extensible Authentication Protocol (EAP). Questo standard può prevenire punti di accessi anomali e altri dispositivi non autorizzati dal divenire backdoors non sicure.

“Don’t use Microsoft Internet Explorer, Outlook, Outlook Express, Microsoft Office and unistall the Windows Scripting Host. If possible, don’t use Microsoft Windows, use Linux.”

Bruce Shneier (cryptographer, computer security professional and privacy specialist)

Dimostrazione correttezza RSA

Dati
n = p * q
e t.c. GCD(e, φ(n)) = 1
d t.c. e * d mod φ(n) = 1

Funzioni
C(m) = $m^e \ mod \ n$ = c
D(c) = $c^d \ mod \ n$ = m

Dimostrazione di correttezza
Si vuole dimostrare che *D(C(m)) = m*

- Prerequisiti
Teorema di Eulero
Dati m ed n, se essi sono coprimi, ovvero GCD(m, n) = 1, allora $m^{\varphi(n)} \bmod n = 1$

Teorema del resto cinese
Dati $m^k \bmod p = 1$ e $m^k \bmod q = 1$, allora $m^k \bmod pq = 1$

Proprietà dell'aritmetica modulare
Se $m \bmod n = 1$ allora $m^k \bmod n = 1$, e anche se $m \bmod n = 0$ allora $m^k \bmod n = 0$

- Dimostrazione
$D(C(m)) = (C(m))^d \bmod n = (m^e \bmod n)^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$

Per come sono costruiti *d* ed *e*, si ha che $ed \ mod \ \varphi(n) = 1$. Questo significa che esiste un *k* tale per cui
$ed = k\varphi(n) + 1$. Sostituendo:

$m^{ed} \bmod n = m^{k\varphi(n) + 1} \bmod n = m * m^{k\varphi(n)} \bmod n$

Assumendo che *m* ed *n* siano coprimi, si ha che GCD(m, n) = 1 e quindi per il teorema di Eulero (che si assume già dimostrato) $m^{\varphi(n)} \ mod \ n = 1$. Noi siamo in presenza però di un *k*, ma grazie alle proprietà dell'aritmetica modulare $m^{\varphi(n)} \ mod \ = m^{k\varphi(n)} \ mod \ n = 1$.

$m * m^{k\varphi(n)} \bmod n = m * 1 = m$ C.V.D.

- Note

Nel caso in cui ne *p* ne *q* siano divisori di *m* (ed entrambi primi) allora il loro prodotto $pq = n$ è coprimo con *m*, ovvero GCD(n, m) = 1 (come abbiamo assunto nella dimostrazione).

Nel caso in cui sia *p* che *q* siano divisori di *m* (ed entrambi primi), *pq* è anch'esso un divisore di m, tuttavia questo è impossibile in quanto m < n.

Nel caso in cui *p* divide *m* e *q* non lo divide (ed entrambi primi), dato che *q* è primo e non divide m si avrà
GCD(m, q) = 1, per il teorema di Eulero $m^{\varphi(q)} \bmod q = 1$ e quindi per le proprietà dell'aritmetica modulare moltiplicato per un qualsiasi valore sarà sempre uguale ad 1, questo valore può anche essere $\varphi(p)$ e avremo allora $m^{\varphi(q)\varphi(p)} \bmod q = m^{\varphi(n)} \bmod q = 1$.
Esisterà comunque un intero *k* tale che $m^{\varphi(n)} = kq + 1$, moltiplicando entrambi i membri per un certo *m* che è uguale a p^*c (con *p* numero primo iniziale e *c* valore che gli permette di eguagliare *m*), questo non incide sul risultato ma avremo:
$mm^{\varphi(n)} = kqm + m \rightarrow$ ma sapendo che $m = pc \rightarrow mm^{\varphi(n)} = kqpc + m \rightarrow$ nota che $q^*p = n \rightarrow$
$mm^{\varphi(n)} = kcn + m$ che in modulo sarebbe $mm^{\varphi(n)} \bmod n = m$ come volevamo.

Nel caso in cui *p* non divide *m* e *q* lo divide, è come il passaggio sopra ma con *p* e *q* invertiti.