Riassunto Network Security and Management

Tommaso Puccetti
Studente presso Universita degli studi di Firenze

November 29, 2018

Contents

List of Tables

List of Figures

1 Security Basics

Propriet della **Security**:

- **Confidentiality:** assicurare che persone non autorizzate accedano alle informazioni.
- **Integrity:** assicurare che le informazioni non vengano alterate da individui non autorizzati, in un modo che non sia individuabile dagli utenti autorizzati
- **Authentication**: Assicurarsi che gli utenti siano chi dicono di essere.

La security non deve essere confusa con la sicurezza. **Security:**

- La qualit o lo stato di essere sicuri (liberi da pericoli, da paura o ansi, libero dalla prospettiva di essere licenziato);
- Qualcosa di dato, depositato o impegnato con lo scopo di rendere un impegno un obbligo;
- Uno strumento di investimento nella forma di un contratto, che fornisce l'evidenza della sua propriet;
- Qualcosa che protegge (misure messe in atto contro lo spionaggio o sabotaggio, crimini o attacchi.)

Per quanto riguarda la safety:

- La condizione di essere sicuri rispetto al subire o causare danno, infortuni, o perdite.
- Un dispositivo progettato per prevenire operazione involontarie o pericolose.

La sicurezza ha un **costo**: un sistema sicuro **pi complesso da realizzare e da manutenere**, in definitiva **pi complesso**. BLABLABLA

2 NAT

Problema: *gli indirizzi IP sono pochi e costosi, per di pi non sempre vogliamo esporre la struttura interna di una Intranet (rete locale).*
Pe questo motivo vengono utilizzate classi di indirizzi IP (IPv4) **non-routable** come definito in **RFC 1918**, riservati alle reti locali con lo **scopo di ridurre le richieste su indirizzi pubblici**. I pacchetti con tali indirizzi per l'instradamento

e l'indirizzamento tramite protocollo IP da router internet.
Si utilizza il **NAT** e **NAPT** mascherano un indirizzo tramite proxy a livello IP:

- Si trasforma un indirizzo sorgente (IP Number e port) in un altro indirizzo.
- Il server NAT viene visto all'esterno come la sorgente della comunicazione
- Il NAT **trasparente** per l'utente interno

Classi di indirizzi privati:1

Class	Private Address Range
A	10.0.0.0 - 10.255.255.255
B	172.16.0.0 - 172.31.255.255
C	192.168.0.0 - 192.168.255.255

Figure 1: Security concepts and relationships

2.1 NAT statico

- Si ha un mapping uno a uno tra indirizzi esterni ed interni.
- Pu essere utilizzato in congiunzione con un firewall.
- Non risolve il problema della scarsit di indirizzi.
- Risulta molto facile da implementare

NAT Statico:2

2.2 NAT Dinamico

- Mapping dinamico tra indirizzi esterni ed indirizzi esterni
- Risolve il problema della scarsit degli indirizzi
- Richiede Server stateful (mantiene informazioni di stato dell'utente durante una sessione).

NAT Dinamico:3

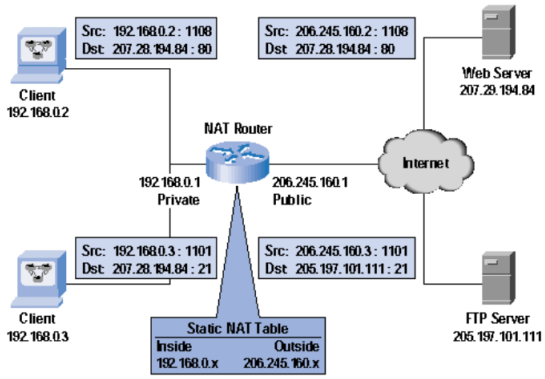

Figure 2: Security concepts and relationships

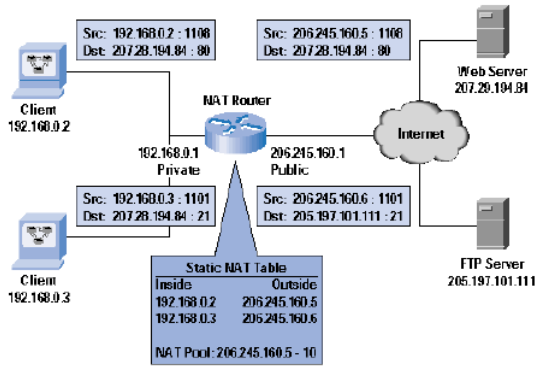

Figure 3: Security concepts and relationships

2.3 NAPT: Network Address and Port Translation

- Mapping dinamico tra indirizzi interni e d esterni **con porte dinamiche**.
- Risolve il problema della scarsit di indirizzi
- Richied un serve stateful pi complesso rispetto al NAT.

NAPT:4

Tuttavia il NAT ha una controindicazione: implica un **ricalcolo dei checksum** IP e TCP come avviene in IPSec (standard per reti a pacchetto per la sicurezza su reti IP). Le due cose possono **interferire portando ad un completo blocco delle comunicazioni**. Si hanno problemi sia con

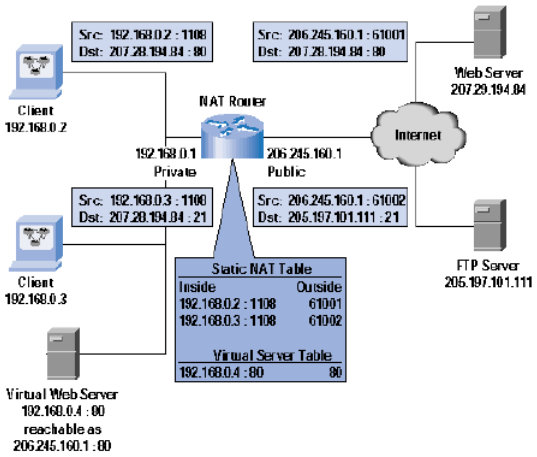

Figure 4: Security concepts and relationships

la funzione **AH** (Authentication Header, protocollo per controllo integrit di pacchetto, garantisce authentication pacchetto per pacchetto tramite checksum a chiave simmetrica) sia con la funzione **ESP** (Encapsulating Security Payload utilizzato per autenitcit confidenzialit e integrit) di IPSEC

INSERIRE natipsec

La soluzione pu essere quella di applicare il NAT e poi IPSEC, o in alternativa eseguirli insieme. Da tenere in considerazione il fatto che un Host dietro a un NAT non pu cominciare una comunicazione IPSec (???????). Inoltre la **co-locazione di NAT e IPSEC un potenziale pericolo per la sicurezza**. La terza opzione quella di utilizzare un tunnel IP-over-IP ma deprecabile.

2.4 Basi

I pacchetti non routable non sono trasportati in internet (ovvero i router li scartano) poich il loro indirizzo IP non univoco. Serve pertanto un traduzione da indirizzi non routable a indirizzi routable: il NAT si occupa di questo.

INSERIRE basi

Un NAT svolge le seguenti **operazioni** sia quando arriva un pacchetto sull'interfaccia **interna** che su quella **esterna**:

- Si cerca un **binding** se c' si trasla il pacchetto e si esegue un **forward** di quest'ultimo, altrimenti si **scarta** il pacchetto
- Allo scadere di un **timer** specifico si **cancella il binding**

2.5 NAT RFC 1631 e RFC 2776

Per quanto riguarda **RFC 1631** si varia **solo gli indirizzi IP**, in questo modo tuttavia non risolviamo il problema della scarsit di indirizzi. Infatti il numero di indirizzi necessari pari al numero di PC che vogliono utilizzare *contemporaneamente* lo stesso protocollo.

INSERIRE 1631

In RFC 2776 invece, si varia **sia indirizzi IP che le porte**. In tal modo il numero delle sessione contemporanee (ovvero il numero di bindings contemporanei) pari circa a 64000 (escludendo le porte ben conosciute.)

2.6 NAT binding

Per binding intendiamo una relazione:

$$IP, proto, port(int) <=> IP, Proto, Port(ext)$$

In realt quello che inteso come bindig composto da **Binding + Filter**.

- Il primo associa indirizzo porta interna a un indirizzo porta esterna (realizza la funzione $interno <=> esterno$)
- Il secondo decide se e quali pacchetti dall'esterno vanno ritradotti. ***Attenzione***: il comportamento del filter genera differenti comportamenti del NAT, alcuni voluti altri no.

Il binding varia a seconda dei protocolli che utilizziamo, nello specifico parliamo delle differenze che si riscontrano tra **TCP** e **UDP** a livello di NAT:

- TCP **stateful**, dunque il binding aggiornato in base ad un timer che varia a seconda dello stato della connessione e della dimensione della CWIN. Per questo protocollo il NAT ha un comportamento **symmetric** ossia **binding e filter sono basati sulla stessa quintupla**

  $$(protocollo, IP, portesorgente - destinazione)$$

  (Quintupla ?????)
  Per questo motivo **le comunicazioni devono partire dall'interno** e non possbili effettuare una callback, quindi PASSIVE FTP (????????). Inoltre il **demultiplexing definito a livello TCP**

- UDP **stateless**, il binding basato solo su un timer e sulla conoscenza del comportamento dell'applicazione (informazioni sulle porte utilizzate ad esempio). Il **demultiplexing** fatto a **livello applicazione**, in questo modo una sola applicazioni pu utilizzare una sola socket in uscita per due stream diversi con destinatari diversi (a differenza di TCP).

Abbiamo bisogno di un comportamento diverso del NAT per UDP. Esistono diversi modi di implementare NAT per UDP, queste diverse implementazioni dipendono dalle modalit di esecuzione del Filter. In base a come si comporta NAT alcuni applicativi possono funzionare o meno, in parte o del tutto.

2.6.1 Symmetric NAT

Funziona esattamente come il symmetric per TCP, non funzioneranno i programmi che hanno bisogno di referral e handover (?????)

INSERIRE sym

2.6.2 Full Cone NAT

Il filter non fa niente. Tutto e tutti potranno raggiungere il sorgente (compresi malintenzionati, permetto perfino di eseguire un **port scanning**)

INSERIRE cone

2.6.3 Restricted Cone NAT

Il **Filter basato sull'IP del destinatario.** Significa che accettiamo comunicazioni da porte diverse purch abbiano lo stesso IP (provengano dallo stesso Host). **Non c' controllo sul numero di porta**. Questa politica del Filter restrittiva poich non permette a programmi come MSN e mulo di funzionare

INSERIRE rest

2.6.4 Port Restricted Cone NAT

Il filter basato sulla porta del destinatario. Funzionano tutti i programmi UDP anche se con delle limitazioni.

INSERIRE port

2.6.5 NAT - STUN

Come pu un'applicazione conoscere il tipo di NAT ?
Si utilizza un protocollo chiamato **STUN**, un protocollo **request-reply**. Esso permette alle applicazioni in esecuzione su un computer di scoprire la presenza ed i tipi di NAT e firewall che si interpongono tra il computer e la rete pubblica. Permette inoltre a questi computer di conoscere gli indirizzi IP e le porte con cui il dispositivo NAT li sta rendendo visibili sulla rete pubblica. **Ha a disposizione due porte sul client e due porte e due indirizzi ip sul server**
STUN non grarantisce una conoscenza accurata, infatti il **NAT pu essere non deterministico**, ossia cambiare il comportamento a seconda della disponibilit delle risorse. Un altro problema si riscontra nella possibilit che ci siano pi NAT nel percorso sorgente-destinazione, in questo caso la classificazione non rigorosa e il comportamento imprevedibile (Il secondo livello di NAT potrebbe non avere lo stesso comportamento del primo)

2.7 NAT: ulteriori classificazioni

I NAT possono essere classificati in base a tre parametri:

- Come viene fatto il **binding**.
- Come vengono **aggiornati i filters.**
- Quando si riavviano i **timers.**

2.7.1 In base al Binding

- **Endpoint**: il NAT riusa il binding per tutte le sessioni provenienti da stesso IP/PORTA, IP/PORTA esterni non vengono valutati (**come full cone NAT**)
- **Endpoint**:Il NAT riusa il binding per tutte le sessioni provenienti dalla stesso IP/porta verso lo stesso IP esterno (la porta non si considera). **E come un Restricted Cone NAT.**
- **Endpoint address and port dependent**: come symmetric NAT.

2.7.2 In base al Port Binding

- **Port preservation**: Il NAT tenta di mantenere la porta di origine. Se due Host interni utilizzano la stessa porta di origine uno l'avr cambiata l'altro no.
- **Port overloading**: Il NAT fa port preservation in modo aggressivo, un secondo tentativo di binding fa scadere il binding esistente
- **Port**: ??????

2.7.3 In base al Timer Refresh

- **Bidirectional**: il timer aggiornato dai pacchetti in entrambe le direzioni.
- **Outbound**: Solo pacchetti interno verso l'esterno rinfrescano i timer. Risulta necessario usare un **keep alive**. Inoltre il timer potrebbe essere per session o per binding (nel caso di riuso del binding per piu sessioni)
- **Inbound**: solo i pacchetti dall'esterno verso l'interno rinfrescano il timer, anche in questo caso c' bisogno di un keep- alive
- **Transport protocol state**: come in TCP ma si possono usare altre informazioni (da la possibilit di fare attacchi DOS).

2.7.4 In base all'External Filtering

- **Endpoint independent**: non filtra o scarta pacchetti (full cone)
- **Endpoint address dependent**: Filtra i pacchetti che non provengono dall'IP originario del binding (restricted cone).
- **Endpoint address and port dependent**: Filtra i pacchetti che non provengono dall'IP/porta originario del binding (port restricted cone o symmetric).

2.8 Considerazioni

Per quanto riguarda le **applicazioni p2p** esse tendono ad aggirare il NAT ma cos facendo creano spesso problemi di sicurezza. Per quanto riguarda ICMP rischia di fallire per lo stesso motivo di IPSEC (nel payload sono spesso contenute info su IP e porta originante). Rispetto all'**IP fragmentation** il problema quello di ricostruire i pacchetti (o almeno mantenute informazioni sul primo pacchetto), perch nei frammenti successivi **manca header UDP/TCP**, ma **potrebbe essere un attacco a frammentazione**. Inoltre il primo pacchetto pu arrivare fuori sequenza. Una soluzione quella di provare a configurare il nat in modo che esso stesso modifichi il contenuto del payload.

2.8.1 NAT: UPnP e IGD

Universal Plug n Play: Set di protocolli per la definizione e l'annuncio di device e servizi. Un dispositivo compatibile UPnP pu unirsi dinamicamente ad una rete, ottenendo un indirizzo IP, annunciare il suo nome, trasmettere le proprie capacit su richiesta e venire a conoscenza della presenza e delle capacit degli altri device della rete.
L'**Internet Gateway Device (IGD)** permete ad un device UPnP di scoprire l'indirizzo esterno di un NAT e di creare filters e bindings per i suoi servizi in modo automatico. In questo modo le porte sono aperte in modo incontrollato e potrebbero sovrascrivere i binding esistenti... come per la porta 80 (implementato in Windows).

3 Crittografia

La crittografia la scienza di mantenere segrete le informazioni.
Oggi la criptografia utilizzata, oltre che per la segretezza, per tutti gli altri servizi di sicurezza che abbiamo visto, **esclusa la disponibilit del servizio**. Infatti un uso diffuso delle tecniche di cifratura aumenta le possibilit di incorrere in attacchi **DoS**.

- Protezione documenti:
  - **Integrit**
  - **Segretezza**
  - **Autenticazione**
  - **Non ripudiabilit**
- Verifica identit dei corrispondenti: **controllo degli accessi**.

Vediamo le principali funzioni crittografiche.

3.1 Funzioni Hash

Risolvono il problema della **garanzia di integrit** di un documento trasmesso.
*Una funzione hash una funzione unidirezionale che si applica ad un'informazione, generando un'impronta (**digest**) di dimensione fissa che funzione dei dati in ingresso.* Generalmente sono sequenze di operazioni elementari quali **shift** e **XOR** sui dati, quindi **molto veloci da computare**.

INSERIRE sha

Esempio:

- A invia a B un messaggio M, calcola **H(M) = D** ed invia anche D.
- B riceve M e D, calcola H(M)= D'. **Se D=D'** si ha la garanzia che il messaggio non stato modificato.
- E potrebbe intercettare solo M e modificarlo, ma quando B ricever anche D, l'hash sar diverso e quindi B si accorger che M stato modificato. Per riuscire a modificare M in M' E deve intercettare anche D e cambiarlo in H(M').

Un'applicazione tipica quella della distribuzione di file eseguibili: quando scaricate un eseguibile il file deve essere identico a quello che il produttore ha generato, anche un bit pu provocare un malfunzionamento. Con le ISO degli OS viene fornito il codice **md5** del file che il digest creato con l'hash. Una funzione hash deve avere i seguenti **requisiti minimi**:

- **Compressione:** H mappa un input di lunghezza finita e arbitraria in un output H(x) di lunghezza fissata n.
- **Facilit di computazione:** dato H e un input x, H(x) facile da computare

In aggiunta abbiamo:

- **preimage resistance:** per ogni output computazionalmente infattibile trovare un qualsiasi input che tramite hash genera quell'output. Dato un output y impossibile trovare x tale che H(x)=y.
- **2nd preimage resistance:** computazionalmente infattibile trovare un qualsiasi secondo input che ha lo stesso output di uno specifico primo input. Infattibile trovare x!=x' tale che H(x')=H(x).
- **Resistenza alla collisione:** impossibile trovare qualsiasi coppia di input x e x' che producono lo stesso digest (H(x')=H(x)).

Uno dei **limiti** delle funzioni hash risiede nel fatto che se l'attaccante intercetta il digest pu modificarlo, facendo passare inosservata la modifica al contenuto del file (si possono usare pi funzioni hash). Per questo motivo le hash si accompagnano ad altri metodi di autenticazione.

3.1.1 HMAC

Keyed-hash-message authentication code **accoppia l'utilizzo di una chiave simmetrica ad una funzione hash** per garantire, oltre all'integrit, **l'autenticazione** dei dati. Per **chiave simmetrica** intendiamo una stringa opportunamente lunga scelta in modo meno predicibile possibile. Spesso si utilizzano delle funzioni hash per generare una chiave simmetrica a partire da una data **password**:

$$K = md5(password) = 0x12ab5893092ba4183f3a345872b34f233$$

Se si dispone di una buona funzione hash si pu generare un HMAC componendo la funzione hash con la chiave.

$$HMACK(M) = H((K \oplus opad)_H((K \oplus ipad) \vee M))$$

In questo modo il digest pu essere calcolato solo se si conosce la chiave K.
Passiamo ora a descriverne il **funzionamento:**

- A e B si mettono d'accordo su una password utilizzando un canale sicuro.
- A per inviare un messaggio a B :
  1. calcola K = md5(password)
  2. calcola $HMACK_K(M)$
  3. invia a B la coppia M, $HMACK_K(M)$
- M e $HMACK_K(M)$ possono essere inviate accoppiate nello stesso pacchetto.

INSERIRE hmac

I **vantaggi rispetto ad una semplice funzione hash** sono:

- Se E intercetta sia M che $HMACK_K(M)$ per cambiare M dovrebbe:
  - Modificare M in M'
  - ricalcolare $HMACK_K(M)$
- tuttavia E non in possesso di M (K no?????) quindi non pu calcolare l'HMAC.

Schemi di questo tipo sono utilizzati per garantire integrit e implicitamente anche l'autenticazione di pacchetti livello MAC in molte tipologie di reti.
Di seguito una lista delle principali problematiche:

- **Problema di gestione**: A e B hanno bisogno di una canale sicuro per scambiare la chiave, non utile per comunicazioni via internet.
- **Attacchi di forza bruta**: E potrebbe intercettare una pacchetto e cercare di indovinare la chiave K:

- dato M, e K = 0x00000000000000000000000000000000
- D = HMACK (M) ? se vero allora K la chiave giusta, altrimenti
- K = 0x00000000000000000000000000000001, riprova. . .

Un attacco di questo tipo **computazionalmente impegnativo** per calcolare tutte le chiavi possibili (2^{128}) ci vogliono migliaia di anni. Tuttavia se la chaive generata da una password l'attacco **diventa possibile** utilizzando un **dizionario**:

- dato M e K = md5(abaco), D = HMACK (M)
- se falso, K = md5(abate). . .

Le parole di un dizionario possono essere decine di migliaia, per generarle tutte ci vogliono pochi minuti. ***Per questo le password non devono essere scelte come parole esistenti!***

3.2 Cifratura a chiave simmetrica

Principio di Kerchoff:

- Gli algoritmi crittografici devono essere **noti a priori**.
- Un prodotto che garantisce cifratura con **algoritmo segreto**, non un buon prodotto.

Spieghiamo il **funzionamento** :

- A e B si accordano su una chiave K o una password da cui generare K (come in HMAC).
- I messaggi possono essere decifrati solo con una chiave K.
- Stesso algoritmo per cifrare e decifrare.

INSERIRE des e des2

Tra i **problemi** che si riscontrano si ha la **poca flessibilit** dovuta alla necessit per A e B di scambiarsi le chiavi in anticipo. Inoltre ci espone ad **attacchi di forza bruta** anche se pi difficili del caso HMAC (per E sar pi difficile ad ogni tentativo stabili se il messaggio ottenuto corretto)
Possiamo combinare HMAC e cifratura a chiave simmetrica per ovviare a questi problemi:

1. Si genera HMAC con chiave K
2. Si cifra tutto il pacchetto compreso l'HMAC con una seconda chiave K'.

In questo modo abbiamo segretezza e **integrit** dei dati oltre che ad una forma di **autenticazione** in relazione a come vengono scambiate le chiavi. Questo approccio viene utilizzato per le reti LAN nelle quale si pu impostare a mano sulle macchine.

3.3 Cifratura a chiave pubblica/privata

- A e B possiedono due chiavi ciascuno (possono essere generate insieme da programmi appositi)
  - Una chiave pubblica Pub_A e Pub_B
  - Una chiave privata $Priv_A$ e $Priv_B$
- Le chiavi private si devono tenere segrete (vitale che A sia unico possessore di Pub_A cos per B)
- La chiave pubblica pu essere pubblicata.

Caratteristiche:

- Ci che cifrato con chiave pubblica pu essere decifrato solo con la rispettiva chiave privata.
- Computazionalmente impossibile risalire ad una chiave privata da una pubblica.
- Se A cifra un messaggio utilizzando la chiave pubblica di B solo B potr decifrarlo con la sua chiave privata. **Si ottiene segretezza**.

Non necessario concordare preventivamente una chiave di cifratura comune.

INSERIRE pubpriv

3.4 Firma digitale

Le chiavi pubbliche e private sono invertibili: A pu utilizzare la sua chiave privata per criptare un messaggio che pu essere decriptato utilizzando la relativa chiave pubblica. Il messaggio risulta dunque **decifrabile da chiunque** (la chiave appunto pubblica) dunque **non garantisce la segretezza**. Si pu tuttavia **garantire l'autenticazione e la non ripudiabilit dei dati**: visto che A l'unico in possesso di $Priv_A$ chiunque decifra il messaggio con la sua chiave pubblica **sicuro che il messaggio proviene da quest'ultimo: FIRMA DIGITALE.**

3.4.1 Problemi

Un attacco che possiamo compiere contro questo tipo di cifratura il **Man in the middle**:

- A invia a B Pub_A
- E intercetta il messaggio e scambia Pub_A con Pub_E
- B riceve la chiave Pub_E convinto che sia quella di A e cifra il suo messaggio con Pub_E.
- E intercetta il messaggio, lo decifra, e lo cifra nuovamente stavolta utilizzando Pub_A e lo invia ad A che non si accorge di niente.
- A riceve il messaggio che per pu essere letto e/o modificato da E

Questo tipo di attacco sempre possibile quando A e B non conoscono le rispettive chiavi, che dunque dovranno essere scambiate tramite canale sicuro. Si presenta lo **stesso problema di HMAC e chiave simmetrica**: **l'unica differenza che per sicuro non intendiamo segreto ma semplicemente autenticato**.

3.4.2 Soluzioni: Fingerprint

???????

3.4.3 Soluzioni: Keyserver

Sono **server nei quali possibile effettuare l'upload delle chiavi pubbliche**. Tuttavia il server non assicura la corrispondenza della chiave pubblica all'identit dichiarata. Si possono inserire info come nome e email, dunque facile tramite motore di ricerca trovare la chiave che vogliamo (BUSH)

3.4.4 Soluzioni: Web of Trust

Rete di contatti attraverso la quale i partecipanti certificano l'identit altrui. Il principio di fondo il seguente: se A conosce personalmente B pu certificare che Pub_B ovvero garantire che una certa chiave appartenga effettivamente a B. C, conoscendo A, ha una garanzia maggiore sulla corrispondenza effettiva tra Pub_B e B stesso. Ogni utente ha interesse che la propria chiave pubblica sia certificata dal pi alto numero di persone possibile all'interno della rete. **Le certificazioni sono realizzate utilizzando la propria chiave privata per firmare la chiave pubblica altrui**. Esempio:

- A genera una sua coppia di chiavi PubA e PrivA . Nella ce scritto che la chiave appartiene ad A. chiave pubblica
- A va da B, gli mostra un documento e gli consegna la fingerprint della chiave.
- B scarica la chiave da un keyserver, controlla la fingerprint.
- B a quel punto pu firmare con la propria PrivB la chiave pubblica PubA .
- B carica la chiave firmata sul keyserver.

3.4.5 In pratica: PGP

Pretty Good Privacy stato il primo programma che permetteva di scambiarsi chiavi pubbliche/private e inviarsi messaggi privati. Il governo americano osteggiava la sua distribuzione al pari di tutto ci che utilizzava la crittografia. Nasce come codice sorgente poi chiuso e fatto divenire prodotto commerciale.
Per ovviare a questo, nasce GPG GNU su licenza aperta (GNU GPL).

- c**ome creare chiavi**: gpg gen-key
- **come esportare chiavi**: gpg export armor C93F299D
- **come usare un keyserver:** gpg keyserver pgp.mit.edu send-key C93F299D
- **come cifrare messaggi**: gpg encrypt -r C93F299D file

3.5 Certificati

Le **WOT** sono comode (si basano sul numero di partecipanti, e sulla fiducia reciproca) ma in contesti formali abbiamo bisogno di garanzie aggiuntive, affinch una chiave pubblica sia associata ad una persona. Abbiamo bisogno di **terze parti riconosciute**: **Enti certificatori**.
Questi enti **garantiscono l'associazione tra chiave pubblica e persona fisica**. L'ente ha una sua coppia di chiavi pubblica/privata, gli utenti conoscono quella pubblica. La certificazione avviene come per le chiavi GPG (?????) ma si utilizza un formato diverso (**X.509**).
Come funziona?

- L'utente A genera coppia di chiavi pubblica/privata
- A invia all'ente la chiave pubblica e un documento
- L'ente restituisce la chiave pubblica ad A firmata con la propria chiave privata. **Il contenitore in cui si sposta tale chiave un certificato**.
- La procedura si fa una sola volta, alla creazione della chiave
- L'ente dunque non conosce la chiave privata, **dando cos maggiore garanzia all'utente**. In casi pi semplici la CA fornisce coppia di chiavi e certificato direttamente ad A.
- Quando un utente B deve parlare con A gli chiede il suo certificato.
- Dal certificato estrae la chiave pubblica di A e verifica che la firma digitale dell'ente sia corretta (**utilizzando la chiave pubblica dell'ente certificatore.**) B sicuro dell'identit di A.

La firma digitale ha lo stesso valore probatorio della firma su carta.
Attraverso i certificati otteniamo un accurato controllo degli accessi.

INSERIRE CERTIFICATI

3.5.1 Certificati per tutto

Lo stesso modello pu essere applicato in **qualsiasi contesto** anche senza il bisogno di contattare una CA ufficiale.
Introduciamo l'esempio di un'azienda che ha i seguenti servizi:

- posta elettronica
- sito web
- rete interna a cui collegare pc

Si pu creare una **CA dedicata** con la sua coppia di chiavi pubblica/privata ed un certificato che **autofirmato** (la CA dunque si **autocertifica**). In questo modo si possono rilasciare certificati a tutti gli utenti in modo tale che essi inviino solo posta firmata digitalmente. Ogni volta che un utente riceve (invia ????) una mail questa verr autenticata e cifrata. Inoltre possiamo fornire i browser di certificati cos che i server accetteranno le connessioni solo da macchine autorizzate. Quando un portatile si connette alla rete, prima di essere abilitato a ricevere e inviare traffico, dovr autenticarsi con un server utilizzando un certificato valido.
Un CA autocertificata aumenta il livello di sicurezza, ma dobbiamo sottolineare che se per qualche motivo il server su cui risiedono le chiavi pubbliche e private della CA, **verr compromessa la sicurezza di tutta la rete**.

3.5.2 Certificate Revocation List: CRL

Un utente ha perso un portatile con un certificato valido ? Uno dei server con certificato viene compromesso? Un utente si comporta male ? **Dobbiamo riuscire ad invalidare i certificati gi rilasciati** di fatto revocandoli.
La CRL **una lista di certificati revocati** dall'ente certificatore. Per fare ci la CA tiene una lista di certificati non pi validi, distribuita **firmata con la propria chiave privata**. Un utente deve essere in grado di **scaricare la CRL pi aggiornata** tramite servizio offerto dall'infrastruttura di rete.

Le CRL sono proprio il motivo per il quale abbiamo bisogno di un'autorit di terzi, poich introducono un elemento di complicazione in pi.

3.5.3 WOT di Thawte (tybyca)

COMPLETARE

3.5.4 Confornti e sistema misto

INSERIRE confronto
Le due tecniche vengono utilizzate insieme per garantire performance e sicurezza:

- $A \rightarrow B$: certificato A, C_A
- $A \leftarrow B$: certificato A, C_B
- $A \rightarrow B$: A genera un numero casuale R e invia tale numero cifrato con il certificato ci C_B
- $A \leftarrow B$: B genera un numero casuale P e trasmette tale numero cifrato con il certificato di C_A
- si genera una chiave segreta $K = hash(P \oplus R)$

Una volta generato K possiamo utilizzare quella chiave per cifrare, decifrare e autenticare tutti i messaggi tra A e B. In questo modo abbiamo vantaggi dal punto di vista computazionale.

3.6 Teoria: principi di crittografia

3.6.1 RSA

- Dati p,q primi con $n = pq$
- Dato $m < n$
- se e,d sono inversi moltiplicativi mod $\phi(n)$ ovvero $ed = 1 mod \phi(n)$ allora:

INSERIRE FORMULA EULERO

- Dato $m^e = c$ computazionalmente oneroso ricavare m, per numeri grandi impossibile. **Si pu cifrare esponenziando per e e decifrare esponenziando ancora per d**
- dunque la coppia e,n la chiave pubblica mentre la coppia d,n la chiave privata.
- per valori di e,n non piccoli (maggiori di 1kbit) non esistono algoritmi che permettano:
  - dati e,n ricavare d
  - dato m^e ricavare m

La sicurezza di RSA si basa tutta sull'impossibilit di derivare $\phi(n)$ o d da e,n. Entrambi questi problemi hanno complessit equivalente a quella di fattorizzare n nei suoi fattori primi. Si riuscito a fattorizzare numeri da 332 a 663 bit (con un costo in tempo di mesi), pertanto si ritiene sufficiente una grandezza di 1024 bit per le chiavi.
La sicurezza di RSA dipende tutta dallo stato dell'arte nel campo degli algoritmi di fattorizzazione e nel campo della potenza computazionale.
Un attacco possibile **timing attack** che viene portato in atto nella fase di esponenziazione, ovvero nella fase di decriptazione eseguita tramite la chiave privata. Si basano sul fatto che le operazioni in hardware hanno costi differenti se il bit usato per l'esponenziazione 0 o 1. **Possiamo dunque risalire alle chaive privata solo osservando i tempi di esecuzione della CPU per le operazioni di decodifica**. L'attacco laborioso ma proveniente da una direzione inattesa. Per ovviare a ci le implementazioni di RSA introducono dei ritardi casuali nell'esponenziazione per rendere impredicibili i tempi di esecuzione. Altri algoritmi che utilizzano problemi computazionalmente intrattabili:

- Diffie-Hellman genera una chiave segreta condivisa a partire da due chiavi pubbliche senza bisogno di scambiare alcun segreto
- ElGamal schema basato su logaritmi discreti
- DSA schema a firma digitale basato su logaritmi discreti.

3.6.2 Diffie Hellman

Si basa sull'intrattabilit del logaritmo discreto.
Dato un numero primo p si definisce radice primitiva di p un numero α per cui vale:

$$\alpha mod p \neq \alpha^2 mod p \neq \alpha^3 mod p \neq ... \neq \alpha^i mod p$$

$$i < p$$

Nello scambio entrambi A e B posseggono due parametri noti p,α ciascuno genera un numero casuale X_a e A_b

1. A invia a B $Y_a = \alpha^{X_a} mod p$
2. B riceve Y_a ed invia ad A $Y_b = \alpha^{X_b} mod p$
3. A calcola $K_a = Y_b^{X_a} mod p = (\alpha^{X_b}) X^a mod p = \alpha^{X_b * Xa}$
4. B calcola $K_b = Y_a^{X_b} mod p = (\alpha^{X_a}) X^b mod p = \alpha^{X_a * Xb}$
5. Risulta $K_b = K_a$ dunque A e B si sono scambiati una chiave segreta **senza avere nessuna credenziale comune**

Un attaccante che vede passare solo Y_a e Y_b non in grado di calcolare il logaritmo discreto quindi non pu ricavare la chiave
Diffie Hellman non sicuro contro attacchi **man in the middle**:

- D genera $Xd1$ e $Xd2$ e le chiavi pubbliche corrispondenti $Yd1$ $Yd2$
- A invia Ya a B
- D intercetta il messaggio e trasmette invece $Yd1$ a B
- B riceve $Yd1$ e calcola $K_b = Y_{d1}^{X_b} mod p$
- B invia Yb ad A
- D intercetta Yb e invia invece $Yd2$ ad A
- A calcola $K_a = Y_{d2}^{X_a} mod p$

INSERIRE diffie

3.6.3 Algoritmi a chiave simmetrica: One time pad

1. Dato un testo in chiaro m di n bit si scegli una chiave k generata attraverso un generatore perfetto di numeri casuali
2. La cifratura $c = m \oplus k$
3. ad ogni trasmissione si deve cambiare k

Con questo algoritmo si scorrela completamente c da m, con il risultato che **impossibile effettuare crittanalisi**. Di contro abbiamo che dobbiamo trasportare una chiave k su canale sicuro (della stessa lunghezza di n). In linea teorica questo algoritmo garantisce la **sicurezza pi elevata**, ma nella pratica **difficile da utilizzare**

3.6.4 Algoritmi a chiave simmetrica: Cifratura di Feitsel

La cifratura di Feitsel un algoritmo di sostituzione ideale che mappa un messaggio in chiaro di 2 bit in un messaggio cifrato di 2 bit **utilizzando una mappa statica**.

INSERIRE static

Se il messaggio lungo 2 bit e la mappa 2^2 righe, **l'attaccante pu tentare solo attacchi di forza bruta**, non esiste correlazione statica tra testo in chiaro e testo cifrato. Quando il messaggio pi lungo si cifrano blocchi di 2 bit per volta.
Per risultare sicuro i blocchi devono essere di grandi dimensioni, ma in tal caso **la mappa diventa molto grande** ($n = 64 \rightarrow 64x2^{64} = 2^{70}bit$). Procedimento:

1. Dalla chiave K si generano una serie di sottochiavi K_i con una funzione generatrice
2. Si eseguono una serie di fasi di cifratura che hanno come parametro K_i e il risultato della fase precedente.
3. Tutti gli algoritmi a blocchi come AES utilizzano questo schema di cifratura

INSERIRE feit

3.6.5 Altri algoritmi : DA FINIRE

3.7 Esempio: organizzazione rete certificati

La rete composta da:

- La vostra rete composta da
- Una rete di computer fissi per i vostri utenti
- Un server per applicativi web necessari ai vostri utenti
- Un server per la posta elettronica
- Dei possibili utenti remoti, roadwarrior.

INSERIRE rete

I **servizi** che la rete offre sono:

- non sia possibile collegarsi alla rete interna se non utilizzando i client fissi della rete,
- che ad ogni client della rete possa effetturare laccesso solo personale autorizzato
- che i servizi del vostro webserver siano accessibili solo dai terminali della rete
- che la posta elettronica sia certificata
- che gli utenti possano inviare e ricevere posta elettronica in modo sicuro
- che tutto quello che pu fare un utente dal proprio terminale interno lo possa fare anche un utente roadwarrior