





# SICUREZZA INFORMATICA

Il concetto di **sicurezza** può essere definito come tutte quelle situazioni che possono portare alla perdita di qualcosa di quantificabile che arrecano danno alla vittima.

La sicurezza è definita sulla base dell'asset cioè quello che dobbiamo proteggere valutando quali sono i beni e il loro valore in modo da scegliere il sistema di sicurezza più adeguato. Il sistema di sicurezza viene valutato analizzando i rischi in cui si può incorrere e le conseguenze che derivano nel caso si è violati.

L'asset è tutto ciò che può essere monetizzato e che deve essere valutato sulla base del valore, criticità, insostituibilità, mission. I responsabili della sicurezza sulla base dell'asset, dei rischi adottano delle contromisure adeguate a proteggerlo perché sarebbe inutile spendere troppo per qualcosa che ha poco valore o ha poco probabilità che sia a rischi.

**Safety**-> intendiamo alla protezione dea eventi che possono creare malfunzionamenti che possono arrecare danno a persone o all'ambiente. Si prevengono errori tecnologici.

≠

**Security**-> con cui si intende la protezione da minacce e attacchi intenzionali da persone che vogliono arrecare danno

**Cyberspace**-> è l'ambiente virtuale in cui avvengono le attività informatiche, può essere intesa come l'insieme delle reti e dei sistemi informatici connessi fra loro

|

V

**Information security**-> che sono tutte le misure/controlli per rilevare e prevenire le azioni malintenzionate da parte di utenti non autorizzati

Importante a proteggere i dati perché da questi si possono trarre molte informazioni diverse. Importante è il concetto di **dependability** cioè sulla affidabilità su cui possiamo contare quando usiamo un sistema informatico e quindi sulle capacità che funzioni correttamente in condizioni normali, ma anche di anomalie. Si focalizza sulle:

1. **minacce** quindi le circostanze indesiderate che causano punti di servizio.
2. **attribuiti**, cioè le proprietà che il sistema deve avere.
3. **mezzi** da inviare, cioè, tecniche e strumenti che permettono di avere un servizio affidabile, sicura di qualità.

## ATTRIBUTI

- **affidabilità** → cioè, che il sistema funzioni correttamente senza guasti
- **disponibilità** → cioè, che il sistema sia sempre disponibile
- **sicurezza** → capacità di un sistema di operare in modo da non causare danni alle persone o all'ambiente in caso di guasti
- **manutenibilità** → facilità con cui può essere mantenuto in buone condizioni e con costi contenuti
- **confidenzialità** → restrizioni sui dati
- **integrità** → autenticità dei dati

## MEZZI

- **fault prevention** → come possono essere previsti i malfunzionamenti
- **fault tolerance** → come garantire che un sistema funzioni nonostante i guasti
- **fault removal** → come ridurre il numero e la gravità dei guasti
- **fault forecasting** → come prevedere il numero, la frequenza dei guasti

## MINACCE

- **fault**
- **error**
- **failure**

La **sicurezza informatica** consiste nell'insieme di misure controlli per proteggere i propri **asset** in modo rassicurare confidenzialità, integrità e disponibilità delle risorse del sistema informativo dalle minacce del cyberspazio. Questo perché il **cyberspazio** mette in contatto moltissime reti e persone che possono svolgere molte attività difficili da monitorare per via della eterogeneità dei software e dei protocolli di comunicazione.

Questo rende ogni cosa connessa ad internet potenzialmente attaccabile per via dei punti di debolezza dei sistemi informatici dovuti non tanto ad aspetti tecnologici, ma da aspetti organizzativi e umani. Cioè, spesso si sottovalutano i possibili rischi e si ha un comportamento scorretto della rete da parte dell'uomo.

Però non tutti i punti di debolezza si possono trasformare in **vulnerabilità**, cioè in una falla che può essere sfruttata per fare attacchi, o per corrompere il sistema.

**Attacco**->attività che tenta di arrecare danni o di raccogliere informazioni, di degradare le risorse di un sistema informativo sfruttando le vulnerabilità punto tendono a degradare la CIA.

Gli attacchi sono diventati molto frequenti a causa delle elevate quantità del cyberspazio, ma anche perché mancano le elezioni per internazionali in modo da punire gli attaccanti perché l'attacco può provenire da una qualunque parte del mondo dove la tacco può non essere considerato reato nel paese in cui è avvenuto l'attacco.

Gli attaccanti tendono ad usare tecniche di attacco non fair mentre i difensori per potersi difendere devono seguire la legislazione del loro paese e tendono a nascondere l'attacco subito per motivi di immagine. Gli attaccanti tendono a colpire le infrastrutture critiche dei paesi per creare dei servizi alla gente e creare danni alle istituzioni punto delle aziende tendono a rubare dati sensibili per poterli vendere e puntano sul fatto che non verranno denunciati perché altrimenti l'azienda, se lo mettesse, perderebbe di credibilità.

Oggi la sicurezza informatica coincide anche con la protezione dei dati delle informazioni della persona che la identifica nella società.

**Privacy**->diritto di controllare l'accesso e l'uso delle proprie informazioni personali e alle relative protezioni in modo che malintenzionati non ci accedono.

### Chi attacca?

- **Lamer**, cioè gli inesperti che hanno come scopo imparare oppur si divertono nel causare danni. Non sono pericolosi
- **Hacker** esperto di sicurezza informatica che sfrutta le vulnerabilità per un proprio tornaconto o per fare del bene
  - **Black hat**, sono criminali informatici che violano i sistemi per arrecare danni
  - **White hat**, esperti che usano le loro conoscenze per proteggere i sistemi individuare possibili falle per prevenire gli attacchi
- **Attivisti** individui o gruppi che sono pinti da motivi sociali, politici o ambientali che arrivano danni per promuovere i loro scopi tramite disservizi, furto dati ecc...
- **Organizzazioni sponsorizzate dagli stati** gruppi di hacker supportati dallo stato per vari motivi, come spionaggio

### Come agiscono?

#### 1. Individuano target e raccolta informazioni

Esplora nei siti web aziendali per ottenere varie informazioni sull'obiettivo e raccolgono info sulla rete usando DNS lookup..., mappano la rete con nmap...

Invia un'e-mail per avere informazioni sui clienti di posta elettronica, ma con queste info cercano le vultnerabilità da sfruttare

## **2. Accesso iniziale**

Dammi per forza brutto per ottenere le credenziali di accesso sfruttano le vulnerabilità dei plugin.

## **3. Aumento privilegiato**

Scansione del sistema per individuare exploit locali, cioè, accesso con privilegi limitati e poi si aumentano. Sniffer per ottenere le credenziali da amministratore

## **4. Raccolta di informazioni**

## **5. Mantenimento accesso o copertura**

La sicurezza cerca di proteggere i sistemi alle risorse proteggendo quelli che sono detti i **pilastri della security** e formano la **triade della CIA** (non l'organizzazione di spionaggio americano 😊).

**Confidenzialità** capacità di garantire che le informazioni siano accessibili sui soggetti autorizzati. Riguarda i dati in modo che le informazioni private non vengono divulgate e non si apre una violazione della privacy. Importante proteggerli perché certi dati hanno un elevato valore nel tempo web e sono obiettivi di molti attaccanti.

- **Surface web** è la parte indicizzata dai motori di ricerca e a cui si può accedere ai siti utenti essi
- **Deep web** parte non indicizzata dai motori di ricerca e contiene i dati privati di persone che corrono organizzazioni ecc. E ci si può accedere solo con link diretto e credenziali. Non è illegale.
- **Dark web** è la parte non indirizzata del motore di ricerca e si può accedere solo con software speciali (Tor). Molto spesso è usato per combinazioni illecite.

**Integrità** capacità di garantire che le informazioni non sono modificate o distrutte da soggetti non autorizzati. Non riguarda solo le informazioni ma che i sistemi in cui si deve garantire che funzionino correttamente appunto a questo si aggiunge anche il concetto di

→**Non ripudio**, cioè, la capacità di stabilire chi ha effettuato un'azione senza la possibilità di indicare di averla fatta

→**Autenticità**, cioè la proprietà di essere veri, originali cioè di essere ciò che sia davvero per evitare invasioni o frodi

**Disponibilità** assicurarsi che i sistemi funzionano disponibilità assicurarsi che i sistemi funzionano correttamente e che ci si possa accedere ogni volta che si voglia in un qualunque momento prevede anche correttamente e che ci si possa accedere ogni volta che si voglia in un qualunque momento. Prevede anche

- **Controllo degli accessi** in modo da indicare cessioni che sono autorizzate
- **Resilienza** è capacità di un sistema di continuare ad operare anche quando è sotto attacco e di recuperare l'operatività in un tempo breve.

Gli attacchi cercano di rompere i principi della CIA e possono avvenire a livello hardware, software e reti. Quindi passo al dominio è necessario prendere le particolari contromisure. Sono stati definiti dei principi che non escludono il rischio di malfunzionamento attacco dei sistemi informatici ma che favoriscono la sicurezza.

### **1. Principio di economia del sistema semplicità**

La progettazione delle misure di sicurezza e dell'arte del software deve essere semplice perché in questo modo è più semplice di testare e ricercare possibili vulnerabilità rispetto a un sistema complesso in cui è difficile individuare le debolezze in anticipo

### **2. Principio del default libero da fallimento**

Le decisioni di accesso devono essere basate sui permessi piuttosto che sulle esclusività punto quindi dobbiamo definire le condizioni positive piuttosto che quelle negative perché in questo modo anche un errore di progettazione negherebbe l'accesso.

### **3. Principio di negazione completa**

Ogni accesso deve essere controllato, cioè, il sistema non dovrebbe fare affidamento su informazioni recuperate da cache perché altrimenti devono essere considerati i cambiamenti di privilegio.

### **4. Principio del progetto aperto**

Il progetto di un meccanismo di sicurezza dovrebbe essere libero invece che tenuto segreto in mutui che possa essere attaccato per capire le falle a rendere il sistema più sicuro

### **5. Principio di separazione dei privilegi**

Tecnica con la quale il sistema è suddiviso in più componenti e per ogni componente può essere richiesta per l'accesso privilegi diversi in modo da proteggersi il sistema di azioni in caute o da parte di malintenzionati che potrebbero creare danni permanenti alla macchina se operassero occorrono privilegi pieni.

### **6. Principio del minimo privilegio**

Ad ogni processo/utente. Si devono essere attribuiti i privilegi solo per svolgere le operazioni per cui usano il sistema o le risorse punto in questo modo evitiamo che utenti possano creare danni consistenti

### **7. Principio del meccanismo comune minimale**

la progettazione deve ridurre al minimo le funzioni condivise da diversi

utenti fornendo sicurezza reciproca in modo da ridurre il percorso di comunicazione non voluta

## 8. Principio di accettabilità psicologica

I meccanismi di sicurezza dovrebbero essere semplici da capire in modo che vengono usati meglio e garantiscano protezione questo perché, se fossero troppo complicati gli utenti potrebbero disattivarli o trovare dei metodi per girarli ed usarli in modo superficiale

## 9. Principio di isolamento

I sistemi di accesso pubblico devono essere isolati dalle risorse critiche per prevenire la manomissione/divulgazione.

->isolamento fisico limitando il numero di sistemi che hanno le risorse critiche o assenza di connessione fisica

->isolamento logico si stabiliscono meccanismi di accesso per accesso processi e file degli utenti devono essere separati fra loro, infatti, gli OS forniscono strutture separate per farlo.

Le **superfici di attacco** tramite cui si può ricevere un attacco sono:

- 
- **superfici di attacco di rete**
- **superfici di attacco software**
- **superficie di attacco umano**

**ingegneria sociale** → tecniche sociologiche, psicologiche per manipolare le persone e farle fare quello che serve per raggiungere i propri scopi sfruttando la complessità dell'essere umano agendo sul piano psicologico sfruttando la fiducia che c'è fra i rapporti umani o i **bias cognitivi**, cioè le **devianze** delle persone. Tutti noi possiamo essere target perché possiamo avere denaro o dati che possono essere venduti. Il motivo per cui si spente tempo per trovare tecniche per fregare le persone invece che cercare le vulnerabilità è che la rete **PULLULA DI POLLI DA FOTTERE**, non tutti sono a conoscenza dei rischi che si possono correre in rete.

**Phishing-Social-Baiting-Dumpster Diving-Piggybacking-FakeNews-ClickBaiting-Denigration-Clicktivismo-Grooming-Happy slapping- Outing e trickery-Scam-Dating Online**

In generale per tutti questi attacchi bisogna non credere a tutto quello che troviamo in rete ed essere più critici e non far vincere la curiosità pensando di avere tutto sotto controllo e di saper gestire la situazione perché non conosciamo l'altra parte.



## SOFTWARE MALEVOLO

Le vulnerabilità non si trovano solo nelle persone ma anche nei dispositivi informatici e in particolare nel codice dei software che usiamo punto un software può avere dei bug, errori di implementazione di funzionalità che producono un risultato indesiderato. Il **bug** non è una vulnerabilità voluta perché può nascere dall'inesperienza del programmatore, ad errori di progettazione ecc. I **flaws**, difetti derivanti da una concezione sbagliata del progetto che ha portato sottovalutare certi aspetti. Sono dovuti a specifiche che sono inconsistenti poco dettagliate e ambigue. Sfruttando questi problemi possono essere effettuate attacchi al sistema portando a inserire del codice malevolo, all'interruzione di servizi eccetera.

Il **malware** è un programma che viene inserito in un sistema di nascosto per compromettere la CIA o la disponibilità dei dati/servizi punto questi malware sfruttano le vulnerabilità del codice o del hardware del dispositivo per compiere attacchi che possono essere:

- **passivi**, che non modificano il sistema ma acquisiscono solo le informazioni, configurazione ecc;
- **attivi** che portano a modificare dati, il sistema ecc.

I malware si classificano sulla base dei meccanismi di propagazione di azione, cioè ciò che fanno.

### Meccanismi di propagazione

Infezione e propagazione  
Sfruttamento delle vulnerabilità  
Attacchi basati sull'ingegneria sociale  
Necessità di programma host  
Indipendenti  
replicazione

### Azioni del payload

Corruzione  
Compromissione di servizi  
Furto di informazioni  
occultamento

## KIT DI ATTACCO

Un kit di attacco è un insieme di tool software per sfruttare vulnerabilità includono meccanismi di propagazione metodi di payload che possono essere usati anche dai principianti. Questi si possono modificare, ottenendo qualcosa di più complesso e pericoloso.

## APT

Gli **Advanced Persistent Threats (APT)** non sono semplici malware, ma attacchi informatici sofisticati e prolungati nel tempo, mirati contro obiettivi specifici, spesso aziende o istituzioni politiche.

Questi attacchi sono solitamente condotti da gruppi sponsorizzati da governi o da organizzazioni criminali. Si distinguono per:

- **Advanced (Avanzato):** gli hacker usano vari strumenti e tecniche, compreso malware personalizzato, adattandoli al bersaglio.
- **Persistent (Persistente):** l'attacco dura nel tempo e viene ripetuto finché il bersaglio non viene compromesso.
- **Threat (Minaccia):** è portato avanti da gruppi ben organizzati con risorse e competenze elevate. Il coinvolgimento attivo delle persone nel processo aumenta il livello di minaccia e anche la probabilità di successo dell'attacco.

Gli APT sono tipicamente attribuiti a organizzazioni sponsorizzate da stati nazionali, ma alcuni attacchi possono provenire anche dalle imprese criminali. Gli APT si distinguono dalle altre tipologie di attacco per la loro accurata selezione del target e per i persistenti, spesso furtivi, tentativi di intrusione in periodi di tempo prolungati.

Le tecniche impiegate includono l'ingegneria sociale, le e-mail spear-phishing e i drive-by-download da determinati siti Web compromessi che è probabile vengano visitati dal personale dell'organizzazione target. L'intento è quello di infettare il target con un malware sofisticato utilizzando più meccanismi di propagazione e payload. Una volta ottenuto l'accesso iniziale ai sistemi dell'organizzazione target, viene utilizzata un'ulteriore gamma di strumenti di attacco per mantenere ed estendere tale accesso.

## VIRUS

Un virus informatico è un frammento di software che può “infettare” altri programmi, o comunque qualsiasi tipo di contenuto eseguibile, tramite la loro modifica. La modifica comprende l'inserimento all'interno del codice originale di una procedura per la creazione di copie del codice del virus, che possono a loro volta andare a infettare altri contenuti.

Un virus informatico ha nel codice una procedura per produrre copie di sé stesso. Il virus viene incorporato in un eseguibile di un computer e ogni volta che il computer infetto entra in contatto con un frammento di codice non infetto, una nuova copia del virus viene trasferita nella nuova posizione. In questo modo l'infezione può diffondersi da computer a computer.

Un virus che si inserisce in un programma eseguibile è in grado di fare tutto ciò che il programma è autorizzato a fare. Una volta che il codice del virus è in

esecuzione, esso può svolgere qualsiasi funzione consentita dai privilegi dell'utente che lo esegue.

Un virus informatico è costituito tre parti:

- **Meccanismo di infezione:** il mezzo con cui un virus si diffonde o si propaga, permettendogli di replicarsi. Il meccanismo viene anche chiamato **vettore di infezione**.
- **Trigger:** l'evento o la condizione che determina quando il payload viene attivato o consegnato, talvolta noto come bomba logica.
- **Payload:** ciò che il virus fa, oltre a diffondersi. Il payload può implicare un danno oppure un'attività benigna ma percepibile.

Nel corso della sua esistenza, un virus tipicamente attraversa le seguenti quattro fasi:

- **Fase dormiente:** il virus è inattivo. Al termine di questa fase il virus verrà attivato da qualche evento, come una data, la presenza di un altro programma o file, o il superamento di una certa soglia nella capacità del disco. Non tutti i virus prevedono questa fase.
- **Fase di propagazione:** il virus inserisce una copia di se stesso in altri programmi o in determinate aree di sistema del disco. La copia potrebbe non essere identica alla versione che si sta propagando in quanto i virus spesso mutano per sfuggire al rilevamento. Ogni programma infetto conterrà a questo punto un clone del virus, che a sua volta entrerà in una fase di propagazione.
- **Fase di attivazione:** il virus viene attivato per svolgere la funzione cui è destinato. Come per la fase dormiente, la fase di attivazione può essere scatenata da numerosi eventi di sistema, compreso il conteggio del numero di volte che questa copia del virus ha creato copie di se stessa.
- **Fase di esecuzione:** viene espletata la funzione. La funzione può essere innocua, come un messaggio sullo schermo, o dannosa, come la distruzione di programmi e di file di dati.

I virus svolgono il loro compito in modo particolare al SO o all'hardware che si vuole attaccare, in quanto sfruttano le loro particolari vulnerabilità per svolgere l'attacco. Una volta che il virus ha infettato il sistema, in base ai permessi dell'utente, può infettare gli altri eseguibili.

L'infezione da virus si può prevenire impedendo al virus di entrare, ma è estremamente difficile in quanto un virus può far parte di un qualsiasi programma al di fuori di un sistema. Diverse forme di infezione possono essere impediti negando agli utenti normali il diritto di modificare i programmi del sistema.

Un **macro-virus** è un tipo di malware che infetta i documenti sfruttando le funzionalità di programmazione macro delle applicazioni, come Microsoft Office. Si diffonde attraverso i file anziché i programmi eseguibili, rendendolo particolarmente pericoloso. Sono minacciosi perché:

1. **Indipendenti dalla piattaforma**, quindi, possono colpire qualsiasi sistema che utilizzi applicazioni con macro, senza dipendere dal sistema operativo.
2. **Infettano documenti, non programmi** quindi dato che la maggior parte delle informazioni nei computer è in formato documento, il rischio di infezione è elevato.
3. **Diffusione facile e veloce** quindi si propagano tramite documenti condivisi, spesso via e-mail, e talvolta vengono aperti automaticamente.
4. **Difficili da controllare** poiché infettano file utente e non programmi di sistema, i normali sistemi di sicurezza hanno un'efficacia limitata.
5. **Facili da creare e modificare** perché sono meno complessi rispetto ai virus tradizionali, il che li rende una minaccia costante.

Per proteggersi, è essenziale disattivare le macro nei documenti sospetti e usare software antivirus aggiornati. I macro-virus sfruttano il supporto di contenuto attivo utilizzando un linguaggio di scripting o macro, incorporato in un documento di elaborazione testi o in un altro tipo di file. In genere gli utenti ricorrono alle macro per automatizzare operazioni ripetitive e ridurre così le sequenze di caratteri digitati su tastiera.

Microsoft offre un tool **Macro Virus Protection** che identifica i file Word sospetti e avvisa l'utilizzatore qualora un documento aperto contenesse macro non firmate, o firmate ma non attendibili, e in tal caso veniva loro consigliato di disabilitare le macro. Diversi fornitori di prodotti antivirus svilupparono anche tool in grado di rilevare e rimuovere i macro-virus.

I linguaggi macro possono presentare sintassi simili, i dettagli dipendono dall'applicazione che interpreta la macro, e quindi colpiranno sempre i documenti per un'applicazione specifica.

[MELISSA]

**Classificazione del virus.** Classifichiamo i virus lungo due assi: il tipo di target e il metodo di cui fa uso il virus per nascondersi dal rilevamento.

Una classificazione dei virus per target comprende le seguenti categorie:

- **Boot sector infector:** infetta un master boot record o un boot record e si diffondono quando un sistema viene avviato dal disco contenente il virus.
- **File infector:** infetta i file che il sistema operativo o la shell considerano come eseguibili.

- **Macro-virus:** infetta i file con macro o codice di scripting che viene interpretato da un'applicazione.
- **Virus multipartito:** infetta i file in diversi modi. In genere, il virus multipartito è in grado di infettare più tipi di file, quindi l'eliminazione totale del virus prevede l'intervento in tutti i potenziali luoghi di infezione.

Una classificazione dei virus per strategia di camuffamento comprende le seguenti categorie:

- **Virus criptato:** una forma di virus che utilizza la crittografia per occultare il proprio contenuto. Una porzione di virus crea una chiave crittografica casuale e cifra il resto del virus. La chiave viene memorizzata assieme al virus. Quando viene invocato un programma infetto, il virus utilizza la chiave casuale memorizzata per decifrare il virus. Quando il virus si replica, viene selezionata una chiave casuale diversa. Dal momento che buona parte del virus è cifrata con una chiave diversa per ogni istanza, non è possibile osservare un pattern di bit costante.
- **Virus furtivo:** una forma di virus espressamente progettata per sfuggire al rilevamento dei software antivirus. Ne consegue che l'intero virus, e non soltanto un payload, viene nascosto. Può utilizzare mutazione del codice, compressione o tecniche di rootkit per raggiungere tale scopo.
- **Virus polimorfo:** una forma di virus che nel corso della replicazione crea copie funzionalmente equivalenti, ma con pattern di bit nettamente diversi al fine di eludere i programmi che eseguono scansioni antivirus. In questo caso, la "firma" del virus sarà diversa per ogni copia. Per ottenere tale variazione, il virus può inserire in modo casuale istruzioni inutili oppure scambiare l'ordine di istruzioni indipendenti. Un approccio più efficace consiste nell'utilizzare la crittografia. Si segue la strategia del virus criptato. La porzione di virus responsabile della generazione delle chiavi e dell'esecuzione della cifratura/decifratura viene detta **mutation engine**. La stessa mutation engine viene modificata a ogni utilizzo. Per individuarli vengono usate tecniche euristiche, cioè, l'antivirus è un emulatore della CPU che segue il codice, prima che il programma si avvii per riconoscere del codice malevolo. Se lo trova fa l'abort dell'esecuzione. Ovviamente non eseguono tutto il codice perché, l'utente noterebbe un rallentamento nel sistema. Questo fatto può essere sfruttato per mettere il codice malevolo dopo gli N cicli di clock che l'antivirus usa per eseguire il codice.
- **Virus metamorfico:** proprio come un virus polimorfo, anche un virus metamorfico muta a ogni infezione. La differenza risiede nel fatto che un virus metamorfico si ridefinisce completamente a ogni iterazione, utilizzando più tecniche di trasformazione, incrementando di

conseguenza la complessità del rilevamento. I virus metamorfici possono cambiare il proprio comportamento così come l'aspetto

## WORM

Un **worm** è un tipo di malware capace di **autorePLICARSI** e diffondersi autonomamente attraverso reti e dispositivi, senza bisogno di un'azione da parte dell'utente. Dopo l'attivazione, il worm può replicarsi e propagarsi nuovamente e di solito trasporta una qualche forma di payload.

Per replicarsi, un worm si serve di mezzi per accedere ai sistemi remoti come:

- **Posta elettronica o servizio di messaggistica istantanea**
- **Condivisione di file**
- **Funzionalità di esecuzione remota**: un worm esegue una copia di se stesso su un altro sistema utilizzando un'esplicita funzione di esecuzione remota o sfruttando una falla di programma in un servizio di rete per sovvertirne le operazioni.
- **Funzionalità di trasferimento o di accesso remoto ai file**: un worm utilizza un servizio di accesso remoto ai file o di trasferimento a un altro sistema per copiarsi da un sistema all'altro, dove gli utenti di quel sistema potrebbero poi eseguirlo.
- **Funzionalità di login remoto**: un worm accede a un sistema remoto come utente e poi utilizza dei comandi per copiare se stesso da un sistema all'altro, dove poi viene eseguito

Un worm presenta le stesse fasi di un virus informatico: dormiente, di propagazione, di attivazione e di esecuzione, ma nella fase di propagazione cerca sugli altri sistemi i meccanismi di accesso da infettare (**scanning**) esaminando le tabelle degli host, le rubriche, gli elenchi di amici, i peer attendibili e altri archivi simili contenenti dettagli di accesso al sistema remoto e li sfrutta per trasferire una copia di se stesso al sistema remoto e fa sì che la copia venga eseguita.

Il worm può anche cercare di stabilire se un sistema è stato precedentemente infettato prima di copiarsi su di esso nascondendosi assumendo il nome di un processo di sistema o utilizzando qualche altro nome che potrebbe essere ignorato dall'utente.

## Strategie di scanning degli indirizzi di rete

**Casuale:** ogni host compromesso analizza indirizzi casuali nello spazio degli indirizzi IP servendosi di un seme diverso. Questa tecnica produce un elevato volume di traffico Internet che può causare disservizi generalizzati anche prima che l'attacco vero e proprio venga lanciato.

**Hit-list:** l'attaccante redige anzitutto una lunga lista di macchine potenzialmente vulnerabili. Una volta che la lista è stata stilata, l'attaccante inizia a infettare le macchine presenti nella lista. A ogni macchina infettata viene fornita una porzione della lista da scansionare. Questa strategia si traduce in un processo di scanning piuttosto rapido che potrebbe rendere difficile rilevare che l'infezione è in corso.

**Topologica:** questo metodo utilizza le informazioni contenute nelle macchine vittime infettate per trovare altri host da scansionare.

**Sottorete locale:** se un host può essere infettato alle spalle di un firewall, tale host poi cerca target nella propria rete locale. L'host utilizza la struttura dell'indirizzo di sottorete per trovare altri host che altrimenti sarebbero protetti dal firewall.

[MORRIS]

I **mobile code** come programmi (ad esempio, script, macro, o altre istruzioni portatili) che possono essere trasmessi invariati a un insieme eterogeneo di piattaforme ed eseguiti con identica semantica.

Il mobile code viene trasmesso da un sistema remoto a un sistema locale e poi eseguito sul sistema locale senza l'esplicita preparazione dell'utente. Spesso funge da meccanismo per la trasmissione malware, in altri casi, invece sfrutta delle vulnerabilità realizzare un exploit. I vettori più comuni di mobile code includono le applet Java, ActiveX, JavaScript e VBScript.

Le modalità più diffuse di utilizzo del mobile code per attività dannose sul sistema locale sono: cross-site scripting, siti web interattivi e dinamici, allegati di posta elettronica e download da siti non attendibili o di software non attendibile.

## VULNERABILITÀ LATO CLIENT

I drive-by-download consistono nello sfruttare le vulnerabilità dei browser e dei plugin così che quando l'utente visualizza una pagina web controllata dall'attaccante, tale pagina contiene il codice che sfrutta il bug per scaricare e installare il malware sul sistema senza che l'utente ne sia a conoscenza o abbia dato il proprio consenso. Questo è possibile perché per il funzionamento del protocollo HTTP, le esecuzioni delle risorse che il client chiede non vengono più

eseguite sui server (scalabilità), ma vengono eseguite su host del client. Queste risorse oggi non sono più pagine statiche, ma hanno anche del codice attivo. Questo può essere sfruttato per installare codice malevolo nel host degli utenti. Per difenderci da questo attacco, bisognerebbe aprire le risorse, visualizzando il codice ed impedendo al browser di effettuare il rendering di queste.

Gli attacchi **watering hole** sono una variante dei **drive-by-download**, utilizzata per colpire obiettivi specifici. L'attaccante identifica i siti web che le vittime visitano abitualmente, compromette quelli con vulnerabilità e attende che le vittime li visitino per infettare i loro sistemi. Il codice malevolo può essere programmato per colpire solo determinati utenti, riducendo il rischio di scoperta.

Il **malvertising** sfrutta invece annunci pubblicitari contenenti malware. Gli attaccanti acquistano spazi pubblicitari su siti web mirati e inseriscono codice dannoso che può infettare i visitatori. Questo metodo è efficace perché non richiede la compromissione diretta dei siti e può essere reso più difficile da rilevare generando codice malevolo in modo dinamico o limitandone la diffusione a determinati orari e utenti.

## CORRUZIONE DEL PAYLOAD

Alcuni malware non hanno un payload inesistente il loro obiettivo è quello di nascondersi per ricavare delle informazioni, ma a volte possono avere uno o più payload che svolgono attività segrete a beneficio dell'attaccante.

Ad esempio, la distruzione dei dati sul sistema infetto al verificarsi di certe condizioni di attivazione, mostrare messaggi o contenuti indesiderati sul sistema dell'utente al momento dell'attivazione ecc.

Il **virus Chernobyl** è virus parassita distruttivo, residente in memoria, per sistemi Windows 95 e 98. Infetta i file eseguibili alla loro apertura e cancella i dati sul sistema infetto sovrascrivendo il primo megabyte del disco rigido con degli zeri danneggiando l'intero file system. Se l'attacco ha esito positivo, il processo di avvio fallisce e il sistema diviene inutilizzabile fino a quando il chip BIOS non viene riprogrammato o sostituito. Questi sistemi operativi non avevano un meccanismo di gestione dei permessi degli utenti quindi chiunque eseguisse un programma aveva gli stessi privilegi dell'amministratore del sistema.

Il BIOS, oggi, è stato sostituito da **UEFI (Unified Extensible Firmware Interface)**, che include funzioni di sicurezza avanzate come il **Secure Boot** che impedisce l'esecuzione di software non autorizzato all'avvio del sistema, rendendo impossibile la sovrascrittura del firmware da parte di un virus.

Il worm di **mass-mailing Klez** è un worm distruttivo che ha infettato i sistemi da Windows 95 a XP. Si diffonde inviando copie di se stesso via e-mail agli indirizzi trovati nella rubrica e nei file del sistema. Può arrestare e cancellare alcuni programmi anti-virus in esecuzione sul sistema. Nelle date di attivazione, ovvero il giorno 13 di diversi mesi di ogni anno, fa sì che i file sul disco rigido locale vengano svuotati.

In alternativa, alcuni malware criptano i dati dell'utente e richiedono di pagare un riscatto per ottenere la chiave necessaria al recupero di tali informazioni (**ransomware**).



### PACKET SNIFFING

Lo **sniffing** consiste nell'intercettazione dei dati che transitano in una rete: può essere svolta sia per scopi sia per scopi illeciti per rubare informazioni. Ogni volta che un dispositivo comunica sulla rete, i dati vengono divisi in piccoli pacchetti che vengono trasmessi tra i dispositivi. Quando un attaccante utilizza uno sniffer, questo programma è in grado di **catturare** questi pacchetti di dati in transito sulla rete. Per fare sniffing, l'attaccante deve posizionarsi nel punto giusto della rete per intercettare il traffico.

A seconda del tipo di rete, ci sono diversi modi in cui un attaccante può intercettare i pacchetti, su **una rete cablata (Ethernet)** l'attaccante deve essere connesso fisicamente alla stessa rete locale; su **una rete wireless** l'attaccante può facilmente posizionarsi nei pressi di una rete wireless non protetta. Lo sniffing può avvenire ascoltando solo il traffico di rete senza inviare pacchetti per raccogliere solo i pacchetti di dati o anche **interferendo** o manipolare il traffico, inviando pacchetti falsificati o cercando di dirottare le comunicazioni. Una volta che i pacchetti di dati vengono catturati, l'attaccante può utilizzarli per analizzare i dati. I pacchetti catturati possono essere letti e **decodificati** se non sono criptati o se l'attaccante ha accesso alla chiave di cifratura.

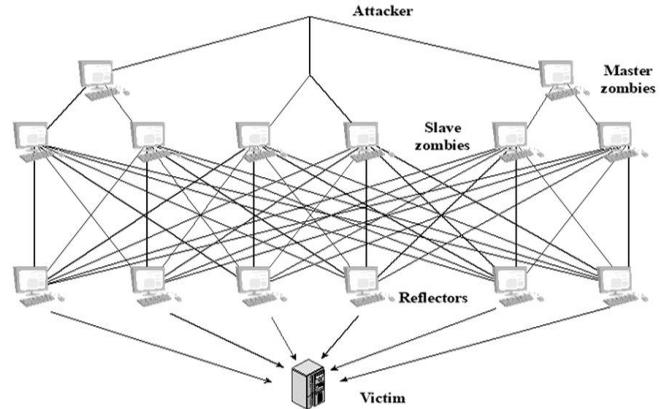
### SPOOFING

Lo **spoofing** è una tecnica utilizzata dagli attaccanti per mascherare la propria identità o per ingannare il sistema in modo che creda che un'origine di dati sia quella legittima, quando in realtà proviene da una fonte non autorizzata. Consiste nel cambiare l'indirizzo IP sorgente di un datagram e farlo apparire come se i dati associati siano stati spediti da un host con un indirizzo IP arbitrario. È necessario modificare il software del protocollo a livello del sistema operativo

## DoS

Un attacco **DoS (Denial of Service)** è una tecnica informatica utilizzata per sovraccaricare un sistema, server o rete, rendendoli inaccessibili agli utenti legittimi. L'obiettivo è

- **saturare la banda** l'attaccante invia un traffico elevato in modo da occupare tutta la larghezza di banda disponibile, impedendo il normale accesso degli utenti.
- **esaurire le risorse** il server viene inondato di richieste, consumando RAM, CPU e altre risorse, fino a bloccare il sistema.
- **sfruttare le vulnerabilità** alcuni attacchi DoS sfruttano bug nei software per causare crash o malfunzionamenti. Questo viene realizzato con malware che prendono il controllo di una macchina, **bot** (robot) o **zombie**, che poi viene utilizzata per lanciare o gestire attacchi che sono difficili da ricondurre al creatore del bot stesso.  
centinaia o migliaia di computer appartenenti a un botnet (botnet) è in grado di agire in modo sincrono per un attacco. Il motivo della botnet è che è più difficile essere individuati quando si vedono arrivare troppe richieste. Un server in grado di riconoscere l'attacco DoS e di reagire in modo adeguato può evitare il disservizio. Un altro motivo è quello di nascondere, perché usa un **handler**: questo programma comanda i dispositivi zombie della botnet. È più difficile risalire a lui. Alcuni attacchi DoS, usano **attacchi di riflessione** che consistono nello sfruttare una vulnerabilità nelle configurazioni di rete e nei protocolli di comunicazione per inviare traffico dannoso verso un bersaglio, senza che l'attaccante stesso debba generare direttamente tutto il traffico, aumentando l'effetto dell'attacco.



## DoS: SYN FLOODING

Un server alloca e inizializza le variabili e i buffer della connessione in risposta a un SYN ricevuto. Se il client non manda un ACK per completare il terzo passo dell'handshake a tre vie, alla fine scadenza del timer, il server termina la connessione mezza aperta e dealloca le risorse.

Il protocollo di gestione della connessione TCP pone le basi per un classico attacco DoS, chiamato attacco SYN flood (SYN flood attack).

In questo attacco l'aggressore manda un gran numero di segmenti TCP SYN, senza completare il terzo passo dell'handshake. L'attacco può essere amplificato mandando i SYN da più sorgenti e creando così un attacco SYN flood di tipo DDoS (Distributed Denial of Service). Le risorse del server riservate alle connessioni possono esaurirsi velocemente perché allocate alle

connessioni mezze aperte. Se le risorse del server sono esaurite, agli utenti legittimi è negato il servizio.

Una difesa efficace, chiamata **SYN cookie**.

- quando un server riceve un segmento SYN, non sa se il segmento arriva da un utente legittimo o se è parte di un attacco SYN flood. Quindi non crea una connessione TCP mezza aperta, ma crea un numero di sequenza TCP iniziale come funzione hash degli indirizzi IP e numeri di porta di sorgente e destinazione del segmento SYN e una chiave segreta nota solo al server.
- Il server usa la stessa chiave segreta per un ampio numero di connessioni. Questo numero di sequenza iniziale è detto “cookie”.
- Il server manda poi un pacchetto SYNACK con questo numero di sequenza iniziale. L'aspetto più importante è che il server non memorizza il cookie o qualsiasi altra informazione di stato corrispondente al SYN.

Se il client è legittimo risponde con un segmento di ACK. Il server, alla ricezione di questo ACK, ha necessità di verificare se corrisponde allo stesso SYN inviato precedentemente. Il server non ha memoria sui segmenti SYN allora usa il cookie. Per un ACK legittimo, il valore nel campo di acknowledgment è uguale al numero di sequenza del SYNACK (il valore del cookie) più uno. Il server eseguirà poi la stessa funzione hash usando i campi del segmento SYNACK (che erano poi gli stessi del SYN originale) e la chiave segreta. Se il risultato della funzione più uno è lo stesso numero del campo di acknowledgment, il server conclude che l'ACK corrisponde al precedente segmento SYN e quindi è valido. Il server crea quindi una connessione TCP completamente aperta e una socket.

Viceversa, se il client non risponde con un segmento di ACK, allora il SYN originale non ha fatto danni, in quanto non erano state allocate risorse.

## DoS: Smurfing

Lo **smurfing** sfrutta il protocollo di rete **ICMP (Internet Control Message Protocol)**, usato per inviare messaggi di controllo, come i **ping**, tra dispositivi di rete.

Ad ogni nodo connesso in rete è associato un Indirizzo IP univoco. La struttura del protocollo IP non impedisce lo spoofing degli indirizzi IP, questo permette a un qualsiasi nodo di generare un pacchetto con indirizzo IP mittente arbitrario. Di conseguenza è possibile l'invio di un pacchetto IP dal proprio nodo, usando l'indirizzo IP di una seconda macchina presente in rete. Una tipologia di richiesta d'informazione del protocollo di diagnostica ICMP (Internet Control Message Protocol) consente l'invio di richieste di eco (ECHO\_REQUEST) al fine di

- Stabilire se è possibile la comunicazione a livello IP tra due nodi
- Verificare l'esistenza di un nodo in una rete
- Calcolare in termini di tempo la distanza che intercorre tra due nodi

L'attacco Smurf consente l'invio pacchetti ICMP ECHO\_REQUEST (tipo 8), generalmente denominati ping, con indirizzo mittente "spoofato" e corrispondente all'indirizzo IP del "computer vittima" (target), a una serie di gateway.

Il gateway mal configurato riceve il ping e lo instrada a tutti i nodi connessi alla propria sottorete, i quali invieranno un pacchetto di risposta ECHO\_REPLY (tipo 0) all'indirizzo IP mittente. In una rete broadcast a multi accesso centinaia di macchine connesse potrebbero rispondere ad ogni pacchetto.

Gli indirizzi destinati al gateway generalmente sono quelli con la porzione dedicata all'host con i bit tutti ad 1. Ad esempio, l'indirizzo IP di broadcast della rete 10.0.0.0 è 10.255.255.255.

Tale tecnica consente di generare molto traffico, soprattutto considerando che i messaggi ICMP sono incapsulati in un datagram IP, che consente l'inserimento di dati opzionali, e nel caso di richieste eco (ECHO\_REQUEST), tali dati saranno replicati nei messaggi di risposta.

Il nodo vittima (target) riceverà un flood di pacchetti ICMP da numerosi indirizzi IP.

## TROJAN HORSE

Un **Trojan horse** è un programma o una procedura utile, o in apparenza utile contenente codice nascosto che, se invocato, svolge alcune operazioni indesiderate o dannose.

I programmi Trojan horse possono essere utilizzati per compiere in maniera indiretta azioni che l'attaccante non potrebbe svolgere direttamente.

Trojan malevoli che spesso contengono payload come spyware per cercare le credenziali bancarie. Pertanto, gli utenti sono tenuti a prendere precauzioni e a verificare la fonte di qualsiasi software che installano.

I Trojan horse possono rientrare in uno di questi tre modelli:

- Proseguire nell'esecuzione della funzione del programma originale e in più svolgere un'attività malevola separata.
- Proseguire nell'esecuzione della funzione del programma originale, ma modificare la funzione per svolgere attività dannose (ad esempio, una versione Trojan horse di un programma di login che raccoglie le password) o per mascherarne altre (ad esempio, una versione Trojan horse di un programma che elenca i processi che non mostra determinati processi dannosi).
- Svolgere una funzione dannosa che sostituisce completamente la funzione del programma originale.

Alcuni Trojan evitano la richiesta di intervento da parte dell'utente sfruttando alcune vulnerabilità software che consentono la loro installazione ed

esecuzione in automatico. Questa è una caratteristica che li rende simili a un worm, ma a differenza di questo, non si replicano.

## BACKDOOR

Una **backdoor** è un punto di ingresso segreto in un programma che permette a qualcuno di acquisire l'accesso al sistema evitando le procedure di accesso di sicurezza. Le backdoor vengono utilizzate in modo legittimo dai programmatore per eseguire il debug e per testare i programmi. La backdoor è un codice che riconosce una sequenza speciale di input o che viene attivato se eseguito da un certo ID utente o da un'improbabile sequenza di eventi.

Le backdoor si trasformano in minacce quando si sfruttano per acquisire un accesso non autorizzato. Una delle strategie impiegate consistette nell'invio di un falso aggiornamento del sistema operativo che conteneva un Trojan horse che poteva essere attivato da una backdoor e che permetteva di acquisire l'accesso al sistema. (Multics)

In tempi più recenti, una backdoor viene implementata come servizio di rete in ascolto su una certa porta non standard a cui l'attaccante può connettersi e inviare comandi da eseguire sul sistema compromesso.

Si può usare un Trojan che si maschera come un'applicazione legittima, ma eseguendolo apre una backdoor.

Un attaccante può implementare una **backdoor** utilizzando le socket creando una **connessione remota** tra un computer compromesso e il suo sistema.

Questo tipo di attacco è chiamato attacco **reverse shell**. Questo approccio è molto difficile da rilevare perché il sistema compromesso avvia la connessione verso l'attaccante, anziché aspettare che l'attaccante si connetta al sistema.

1. L'attaccante crea un server socket sul proprio sistema che ascolta su una porta specifica.
2. Il sistema compromesso avvia una connessione al server dell'attaccante utilizzando un socket. La backdoor è configurata per connettersi automaticamente a questo server quando viene attivata.
3. Una volta stabilita la connessione, il server dell'attaccante invia comandi al sistema compromesso tramite il socket.
4. Il sistema compromesso esegue i comandi ricevuti e invia indietro i risultati al server dell'attaccante attraverso la stessa connessione di rete.

## ROOTKIT

Un rootkit è un insieme di programmi installati su un sistema per mantenere un accesso segreto ad esso con privilegi di amministratore (o root), nascondendo le prove della propria presenza. Ciò fornisce l'accesso a tutte le funzioni e i servizi del sistema operativo. Tramite l'accesso come root un attaccante

dispone del controllo completo del sistema e può aggiunger o modificare i programmi e i file, monitorare i processi, inviare e ricevere traffico di rete e avere accesso da backdoor quando serve.

Un rootkit si nasconde alterando i meccanismi che monitorano e segnalano processi, file e registri su un computer.

Un rootkit può essere classificato sulla base delle seguenti caratteristiche:

- **Persistente**: si attiva ogni volta che il sistema entra in funzione. Il rootkit deve memorizzare il codice in un archivio persistente e predisporre un metodo in base al quale il codice viene eseguito senza l'intervento dell'utente. È semplice da rilevare, visto che la copia nella memoria persistente può essere sottoposta a scansione.
- **Residente in memoria**: non può sopravvivere a un riavvio. Tuttavia, dato che si trova soltanto in memoria, può risultare più difficile da rilevare.
- **Modalità utente**: intercetta le chiamate alle API e altera i risultati restituiti. **Modalità kernel**: può intercettare le chiamate alle API native in modalità kernel. Il rootkit può anche nascondere la presenza di un processo malware rimuovendolo dalla lista dei processi attivi del kernel.
- **Basato su macchina virtuale**: questo tipo di rootkit installa uno snello virtual machine monitor e poi esegue il sistema operativo in una macchina virtuale sopra di esso. Il rootkit può quindi intercettare e modificare in modo trasparente gli stati e gli eventi che si verificano nel sistema virtualizzato.
- **Modalità esterna**: il malware si trova fuori dalla normale modalità di funzionamento del sistema colpito, nella modalità BIOS o di gestione del sistema, dove può accedere direttamente all'hardware.

## MAN IN THE MIDDLE

L'attacco **man-in-the-middle** è un tipo di attacco in cui un avversario intercetta i messaggi tra due parti e può poi trasmettere il messaggio intercettato o sostituirlo con un altro, senza essere rilevato. Le vittime dell'attacco credono di comunicare direttamente tra loro, mentre in realtà la comunicazione passa attraverso il dispositivo dell'attaccante intermedio.

## Principi dell'attacco

- **Intercettazione e manipolazione**: L'attaccante si posiziona tra due comunicanti, intercettando e potenzialmente modificando tutti i messaggi che passano tra loro.
- **Falsa impressione di sicurezza**: Entrambe le vittime credono di avere un canale di comunicazione sicuro e diretto con l'altra parte, mentre è l'attaccante a gestire la comunicazione tra loro. L'attaccante vede tutto e può modificare tutti i messaggi.

- **Inganno:** L'attaccante fa credere a un utente e a un altro dispositivo (ad esempio, un access point) di parlare l'uno con l'altro.

## BUFFER OVERFLOW

Il **buffer overflow** è una condizione che si verifica quando in un'area di memorizzazione o un buffer viene inserita una quantità di dati in input superiore alla capacità allocata, sovrascrivendo altre informazioni. Questo errore di programmazione può corrompere i dati utilizzati dal programma, causare un inaspettato trasferimento del controllo del programma, violazioni di accesso alla memoria e spesso la terminazione anomala del programma. Quando viene sfruttato intenzionalmente come parte di un attacco, il trasferimento del controllo può avvenire verso codice scelto dall'attaccante, permettendo l'esecuzione di codice arbitrario con i privilegi del processo attaccato.

Per sfruttare un buffer overflow, un attaccante deve identificare una vulnerabilità in un programma e capire come il buffer verrà memorizzato in memoria per corrompere le locazioni adiacenti e alterare il flusso di esecuzione. Questo tipo di attacco è uno dei più comuni e deriva da una programmazione inaccurata delle applicazioni. I buffer overflow possono colpire diverse aree di memoria in un processo:

- **Stack Buffer Overflow (o stack smashing):** Si verifica quando il buffer attaccato si trova nello stack, solitamente come variabile locale nello stack frame di una funzione. L'attacco consiste nel sovrascrivere l'indirizzo di ritorno salvato e il puntatore al frame precedente, che si trovano nello stack frame. Le variabili locali sono di solito allocate nello stack frame in ordine di dichiarazione, scendendo verso il basso nella memoria.
- **Heap Overflow:** Colpisce un buffer situato nella memoria allocata dinamicamente nell'heap. A differenza dello stack, non ci sono indirizzi di ritorno diretti da sovrascrivere facilmente, ma un attaccante può modificare un puntatore a funzione all'interno dello spazio allocato che verrà successivamente chiamato, reindirizzando l'esecuzione al proprio shellcode.
- **Overflow nell'area dati globale:** Riguarda i buffer che si trovano nell'area dati globale (o statica) del programma. Anche qui, i dati possono sovrascrivere un buffer globale e modificare posizioni di memoria adiacenti, inclusi puntatori a funzione che verranno invocati.

Gli attaccanti spesso utilizzano il seguente approccio:

- **Iniezione di Shellcode:** Consiste nel copiare codice in linguaggio macchina (shellcode) nel buffer di destinazione e quindi trasferire l'esecuzione a esso. Lo shellcode è specifico per architettura del processore e sistema operativo, e la sua funzione tradizionale è quella di trasferire il controllo a un interprete a riga di comando (shell), fornendo

all'attaccante gli stessi privilegi del programma attaccato. Lo shellcode deve essere indipendente dalla posizione (non contenere indirizzi assoluti) e non può contenere valori NULL all'interno del codice stesso (poiché le funzioni di manipolazione stringhe del C lo interpretano come terminatore di stringa).

- **NOP Sled:** Per ovviare all'imprecisione dell'indirizzo di inizio dello shellcode, l'attaccante può riempire lo spazio precedente al codice con istruzioni NOP (No Operation). Se l'indirizzo di ritorno punta a qualsiasi punto all'interno di questa sequenza di NOP, il calcolatore eseguirà le istruzioni NOP fino a raggiungere l'inizio del vero shellcode.
- **Routine non sicure:** Molti buffer overflow derivano dall'uso di routine non sicure della libreria standard del C, come `gets()`, `sprintf()`, `strcat()`, `strcpy()`, che non controllano la quantità di dati copiati, permettendo la sovrascrittura di variabili adiacenti.

Le conseguenze di un attacco riuscito possono variare da un attacco denial-of-service (DoS) che fa terminare il programma, fino all'elevazione dei privilegi sulla macchina.

## Contromisure e Difese

Le difese contro i buffer overflow possono essere classificate in due categorie principali:

**Difese a tempo di compilazione:** Mirano a prevenire o rilevare i buffer overflow durante la compilazione dei programmi.

- **Scelta del linguaggio di programmazione:** Linguaggi di alto livello con tipizzazione forte come Java, ADA o Python, includono controlli automatici sugli intervalli dei buffer, rendendoli meno vulnerabili.
- **Pratiche di programmazione sicura:** I programmatore devono assicurarsi che il codice che scrive su un buffer controlli sempre che ci sia spazio sufficiente prima di copiare i dati. Si raccomanda di utilizzare routine sicure per la copia di stringhe e buffer (es. `strncpy()` in BSD) o librerie che aggiungono controlli (es. Libsafe).
- **Meccanismi di protezione dello stack:** Estensioni del compilatore che inseriscono codice aggiuntivo all'ingresso e all'uscita delle funzioni:
  - **Stackguard:** Inserisce un valore "canary" (casuale e imprevedibile) nello stack prima delle variabili locali. Questo valore viene controllato all'uscita della funzione; se è stato modificato, il programma viene interrotto. Microsoft Visual C++ include un'opzione /GS con funzionalità simili.
  - **Stackshield e Return Address Defender (RAD):** Copiano l'indirizzo di ritorno in una regione di memoria sicura all'ingresso della

funzione e lo confrontano con quello nello stack all'uscita. Qualsiasi cambiamento provoca la terminazione del programma.

**Difese a tempo di esecuzione:** Implementate come aggiornamenti del sistema operativo, offrono protezione anche per programmi vulnerabili già esistenti.

- **Protezione dello spazio degli indirizzi eseguibili (No-execute bit o NX):** Impedisce l'esecuzione di codice presente in aree di memoria come lo stack o l'heap, etichettandole come non eseguibili. Questo rende inefficaci molti attacchi di buffer overflow che tentano di eseguire shellcode iniettato.
- **Randomizzazione dello spazio degli indirizzi (ASLR - Address Space Layout Randomization):** Rende casuale la posizione di strutture dati fondamentali (come stack, heap e librerie standard) nello spazio degli indirizzi di un processo. Questo rende molto difficile per un attaccante prevedere l'indirizzo di destinazione esatto necessario per un attacco.
- **Pagine di guardia:** Inserisce pagine di memoria illegali tra regioni critiche della memoria di un processo (ad esempio, tra l'area dati globale e altre aree di gestione). Qualsiasi tentativo di accesso a queste pagine provoca l'interruzione del processo, prevenendo overflow in regioni adiacenti.

# CRITTOGRAFIA

La crittografia è una branca della matematica che si occupa della trasformazione dei dati. Gli algoritmi crittografici vengono utilizzati in molti modi nella sicurezza delle informazioni e nella sicurezza della rete. Gli algoritmi crittografici possono essere suddivisi in tre categorie:

- **Senza chiave:** non utilizzare alcuna chiave durante le trasformazioni crittografiche.
- **A chiave singola:** il risultato di una trasformazione è una funzione dei dati di input e un file chiave singola, nota come chiave segreta.
- **A due chiavi:** in varie fasi del calcolo, vengono utilizzate due chiavi diverse ma correlate, denominate chiave privata e chiave pubblica.

## Terminologia

- **Testo in chiaro:** il messaggio o i dati originali che costituiscono l'input all'algoritmo.
- **Testo cifrato:** il messaggio codificato prodotto come output. Tale messaggio
- dipende dal testo in chiaro e dalla chiave. Per un determinato messaggio in chiaro, chiavi differenti produrranno testi cifrati differenti.
- **Algoritmo di decifratura:** operazione inversa dell'algoritmo di cifratura. Prende come input il testo cifrato e la chiave segreta al fine di ottenere il testo in chiaro originale.

## PRINCIPIO DI KERCKHOFFS

La sicurezza di un algoritmo di cifratura non dipende dalla complessità dell'algoritmo di cifratura, ma dal tenere al segreto le chiavi. L'algoritmo può essere reso pubblico in modo che gli attaccanti, attaccandolo, individuino le vulnerabilità e che questi le rendano pubbliche.

Mantenere segreto un algoritmo di cifratura equivale ad avere una **security by obscurity**. Se non si conosce l'algoritmo, i criptoanalisti non possono attaccarlo in maniera diretta, e ciò impedisce di valutare quanto l'algoritmo sia effettivamente robusto.

Per ogni algoritmo di cifratura si definisce il **fattore di lavoro**, cioè la difficoltà computazionale per un criptoanalista nel dover effettuare una ricerca esaustiva nello spazio delle chiavi ammesse per violare l'algoritmo. Il fattore di lavoro è esponenziale rispetto alla lunghezza delle chiavi ammesse.

## **Principi fondamentali di impiego della crittografia**

**Il primo principio di impiego della crittografia** è che tutti i messaggi in codice devono contenere delle informazioni ridondanti, cioè informazioni non necessarie alla comprensione del messaggio in modo che gli intrusi non possano inviare informazioni a caso ed ottenere che queste siano interpretate come messaggi validi

**Il secondo principio** è che non devono mai essere ritenuti validi messaggi recapitati con un qualche ritardo rispetto all'attesa ciò impedisce agli intrusi di intercettare messaggi validi inviati e di poterne ripeterne la spedizione.

L'obiettivo dell'attacco a un sistema crittografico consiste nell'individuare la chiave utilizzata, piuttosto che semplicemente ottenere il testo in chiaro corrispondente a un singolo testo cifrato. Vi sono due approcci generali per attaccare uno schema di crittografia convenzionale:

- **Crittoanalisi:** un attacco di analisi crittografica si basa sulla natura dell'algoritmo e sfrutta qualche conoscenza delle caratteristiche generali del testo in chiaro o eventualmente qualche esempio di coppia testo in chiaro/testo cifrato. Questo tipo di attacco sfrutta le caratteristiche dell'algoritmo per tentare di individuare il testo in chiaro o la chiave utilizzata.
- **Attacco a forza bruta:** l'attaccante tenta ogni possibile chiave su un frammento di testo cifrato finché non riesce a ottenere un testo in chiaro comprensibile.

### **Tipologie di attacchi**

**Attacco con solo testo cifrato:** L'attaccante ha solo il messaggio cifrato e deve analizzarlo tramite test statistici. È il tipo di attacco meno efficace perché fornisce poche informazioni.

**Attacco a testo in chiaro noto:** L'attaccante dispone sia di alcuni messaggi in chiaro che della loro versione cifrata e cerca di individuare la chiave analizzando le trasformazioni applicate.

**Attacco sulle parole probabili:** Se il messaggio contiene termini prevedibili (come intestazioni o frasi ricorrenti), l'attaccante può sfruttare queste informazioni per decifrare il testo.

**Attacco a testo in chiaro scelto:** L'attaccante riesce a far cifrare un messaggio a sua scelta e usa il risultato per cercare di ricavare la chiave.

**Attacco a testo cifrato scelto:** Tecnica meno comune in cui l'attaccante manipola il testo cifrato per dedurre informazioni sulla chiave.

Gli utilizzatori degli algoritmi di crittografia possono desiderare per un particolare algoritmo è che esso soddisfi uno o entrambi dei seguenti requisiti:

- Il costo della violazione del testo cifrato supera il valore delle informazioni crittografate.
- Il tempo richiesto per violare il testo cifrato è superiore al tempo di vita utile delle informazioni.
- Uno schema di crittografia è detto **computazionalmente sicuro** se soddisfa almeno uno tra questi due criteri.

**Attacchi a forza bruta** Un attacco a forza bruta prevede di provare ogni chiave possibile fino a ottenere una traduzione comprensibile del testo cifrato in testo in chiaro. Per avere successo occorre provare la metà di tutte le chiavi possibili. A meno che non si abbia a disposizione il relativo testo in chiaro, l'analista dovrà essere in grado di riconoscere l'effettivo testo in chiaro originale.

**Crittografia robusta** La crittografia robusta fa riferimento a schemi di crittografia che rendono difficilmente raggiungibile l'accesso a un testo cifrato. Le proprietà che rendono un sistema crittografico robusto sono la scelta di un algoritmo di crittografia appropriato, utilizzo di chiavi di lunghezza adeguatamente lunga, scelta attenta dei protocolli, implementazione ben progettata e l'assenza di vulnerabilità introdotte deliberatamente.

---

## CRITTOGRAFIA A CHIAVE SIMMETRICA

---

La crittografia a chiave singola è conosciuta anche come **crittografia a chiave simmetrica** perché la **chiave segreta**, input dell'**algoritmo di cifratura**, è nota sia al destinatario che al mittente della comunicazione. dati. Un algoritmo di decifratura corrispondente prende i dati trasformati e la stessa chiave segreta e recupera i dati originali. La crittografia simmetrica assume le seguenti forme:

- **Cifrario a blocchi:** un cifrario a blocchi opera sui dati come una sequenza di blocchi. La dimensione tipica di un blocco è di 128 bit. Nella maggior parte delle versioni del cifrario a blocchi, note come modalità operative, la trasformazione dipende non solo dal blocco di dati corrente e dalla chiave segreta, ma anche dal contenuto dei blocchi precedenti.
- **Cifrario a flusso:** un cifrario a flusso opera sui dati come una sequenza di bit. In genere, viene utilizzata un'operazione di OR esclusivo per produrre una trasformazione bit per bit. Come con il cifrario a blocchi, la trasformazione dipende da una chiave segreta

## TECNICHE DI SOSTITUZIONE

Una **tecnica di sostituzione** consente di sostituire le lettere del testo in chiaro con altre lettere, numeri o simboli.

**Cifrario di Cesare** Prevede la sostituzione di ciascuna lettera con la lettera che si trova a tre posizioni di distanza nell'alfabeto. L'alfabeto è ciclico e dunque la prima lettera dopo la Z è la A.

in chiaro: a b c d e f g h i j k l m n o p q r s t u v w x y z  
cifrato: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Si assegniamo un numero a ciascuna lettera questa può essere espressa come:

$$C = E(k; p) = (k + 3) \text{ mod } 26$$

L'algoritmo di decifratura è:  $p = D(k; C) = (C - k) \text{ mod } 26$

**Cifrari monoalfabetici** Con solo 25 possibili chiavi, il cifrario di Cesare è tutt'altro che sicuro. Un notevole incremento nello spazio delle chiavi può essere ottenuto consentendo una sostituzione arbitraria. Se invece la riga di testo cifrato potesse essere una permutazione dei 26 caratteri alfabetici, vi sarebbero  $26!$  ovvero più di  $4 \times 10^{26}$  possibili chiavi. Questo approccio è chiamato **cifratura a sostituzione monoalfabetica**, poiché viene utilizzato un unico alfabeto di cifratura (mappatura da un alfabeto in chiaro a un alfabeto cifrato) per messaggio.

Vi è però un altro metodo di attacco. Se l'analista crittografico conosce la natura del testo in chiaro può sfruttare alcune regole del linguaggio come calcolare la frequenza relativa delle lettere e confrontarla con la distribuzione standard delle frequenze per la lingua. Questo se il messaggio fosse abbastanza lungo, altrimenti potremmo avere una corrispondenza sbagliata. Una miglioria consiste nel ricercare la frequenza delle combinazioni di **digrammi** e procedere allo stesso modo.

Le cifrature monoalfabetiche sono facili da violare poiché conservano le informazioni di frequenza dell'alfabeto originario. Una contromisura è quella di utilizzare più sostituti (omofoni) per la medesima lettera.

Un altro modo per migliorare la semplice tecnica monoalfabetica è quello di utilizzare diverse sostituzioni monoalfabetiche via via che si procede nel messaggio in chiaro. Questo approccio viene chiamato **cifratura a sostituzione polialfabetica**.

## TECNICHE DI TRASPOSIZIONE

Si può ottenere una mappatura eseguendo una permutazione delle lettere del testo in chiaro. Questa tecnica è chiamata **cifratura a trasposizione**.

Una tecnica consiste nello scrivere il messaggio in un rettangolo, riga dopo riga, eleggerlo invece colonna dopo colonna, cambiando però l'ordine delle colonne. L'ordine delle colonne diventa quindi la chiave dell'algoritmo. Per esempio:

Chiave: 4 3 1 2 5 6 7

Testo in chiaro: a t t a c k p  
o s t p o n e  
d u n t i l t  
w o a m x y z

Testo cifrato: TTNAAPMTSUOAODWCOIXKNLYPETZ

Per cifrare, iniziamo dalla colonna etichettata col numero 1, che in questo caso è la terza. Quindi, ricopiamo tutte le lettere in questa colonna, procediamo con la quarta colonna che è etichettata col numero 2, e dopo continuiamo con la seconda, la prima, la quinta, la sesta e infine la settima.

La cifratura a trasposizione pura è facile da riconoscere poiché presenta la stessa frequenza delle lettere del testo in chiaro originario. Per il tipo di trasposizione a colonne appena illustrato, l'analisi crittografica è piuttosto semplice e prevede la disposizione del testo cifrato in una matrice, giocando poi sulla posizione delle colonne. Può essere utile anche impiegare tabelle di frequenze di digrammi e trigrammi.

Possiamo dire che un cifrario è sicuro se il testo cifrato "sembra casuale", cioè non dà nessuna informazione sul testo in chiaro. Questa proprietà è chiamata **segretezza perfetta**, ed i cifrari che la raggiungono sono **cifrari perfetti**.

I cifrari a sostituzione non sono perfetti perché mantengono la frequenza delle lettere tipiche della lingua usata e quindi danno informazioni all'attaccante. Bisogna scegliere una parola chiave lunga quanto il testo in chiaro e senza alcuna relazione statistica con esso. Possiamo cifrare

$$c_i = p_i \text{ XOR } k_i$$

dove

$p_i$  = i-esima cifra binaria del testo in chiaro

$k_i$  = i-esima cifra binaria della chiave

$c_i$  = i-esima cifra binaria del testo cifrato

XOR = operatore di XOR

Il testo cifrato viene quindi generato eseguendo l'operazione di XOR bit-a-bit fra il testo in chiaro e la chiave. La chiave, inoltre, deve essere utilizzata per cifrare e decifrare un solo messaggio, ed essere poi scartata. Ogni nuovo messaggio

richiede una nuova chiave lunga quanto il messaggio. Tale schema, chiamato **one-time pad**, è inviolabile in quanto produce un output casuale che non ha più alcuna relazione statistica con il testo in chiaro. Poiché il testo cifrato non contiene alcuna informazione sul testo in chiaro, non vi è alcun modo per violare questo codice.

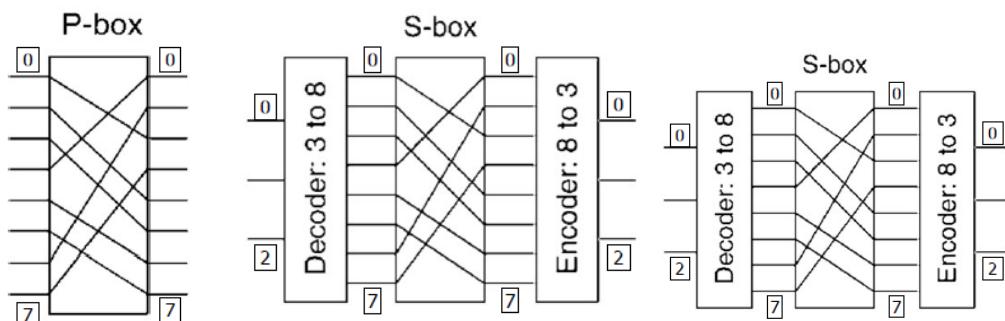
Lo one-time pad è un sistema di crittografia teoricamente inviolabile, in quanto garantisce una sicurezza perfetta perché consiste nell'uso di una chiave segreta completamente casuale, lunga quanto il messaggio da cifrare. Questo significa che ogni bit o carattere del testo viene combinato con un bit o carattere della chiave in modo univoco e irripetibile.

Se un attaccante riuscisse a intercettare il messaggio cifrato, non avrebbe alcun modo di risalire al testo originale, perché ogni possibile chiave potrebbe generare un testo in chiaro plausibile. Non esiste un metodo per determinare quale sia la chiave corretta e, di conseguenza, quale sia il messaggio autentico. Nella pratica presenta delle difficoltà:

1. il problema è creare le chiavi casuali perché la generazione dei numeri casuali si basa su un seed. Usare sempre lo stesso seed comporta la generazione degli stessi numeri ad ogni esecuzione dell'algoritmo.
2. Per ogni messaggio inviato, una chiave di pari lunghezza è necessaria sia al mittente che al destinatario. Pertanto, siamo in presenza di un enorme problema di distribuzione delle chiavi.
3. se il destinatario perde la sincronizzazione dei caratteri del testo ed della chiave per errori in trasmissione o di attacchi, tutti i dati da quel punto in avanti non possono essere decifrati correttamente

Tutti questi problemi hanno portato a realizzare algoritmi di codifica molto complessi che, anche se il criptoanalista entrasse in possesso di una grande mole di testo cifrato, non sarebbe in grado di dare ad esso alcun senso.

Oggi la crittografica moderna si basa ancora sul cifrare a blocchi, realizzati combinando blocchi di sostituzione e permutazione. Non c'è logica dietro queste funzioni, sono solo operazioni fatte con un certo criterio. Il vantaggio è che sono semplici da realizzare anche a livello hardware.



## Cifrari a blocchi

Nella cifratura a blocchi il testo viene diviso in blocchi di N byte che vengono cifrati in maniera indipendente dagli altri blocchi.

Se nel divide in blocchi il testo, se non dovesse coprire gli N bit, viene aggiunto del **padding** per riempire il blocco.

Gli standard **PKCS#5** e **PKCS#7** definiscono un meccanismo di padding

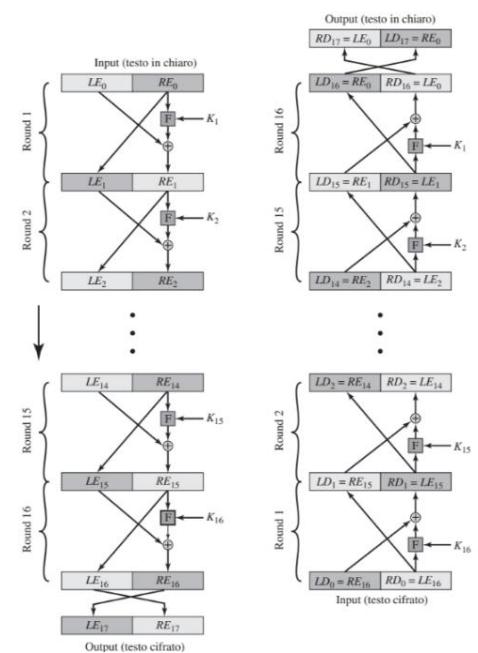
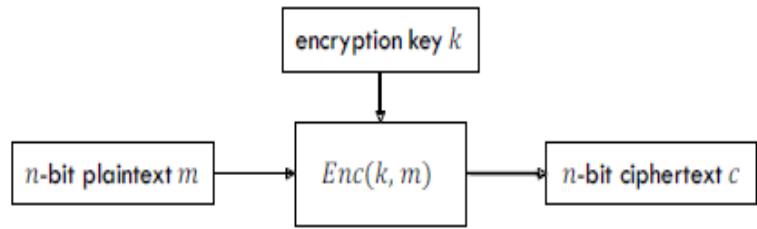
- **PKCS#5** è stato definito per algoritmi a blocchi di 8 byte, come il DES. Si aggiunge il padding, ciascun byte aggiunto assume come valore il numero totale di byte di padding necessari. In questo modo, durante la decifrazione, è possibile identificare con precisione quanti byte devono essere rimossi.
- **PKCS#7**, invece, è una generalizzazione di PKCS#5. Estende lo stesso concetto anche ad algoritmi che utilizzano blocchi di lunghezza diversa, fino a un massimo di 255 byte.

Sono cifrari **monoalfabetici**, cioè, che usano una sola chiave per cifrare tutti i blocchi del plain-text. Questo può essere un problema perché, se si è in possesso di abbastanza testo cifrato è possibile individuare dei pattern ricorrenti e capire quale chiave è stata usata per cifrare il testo.

## DES

Il DES (Data Encryption Standard) è un algoritmo di cifratura a chiave simmetrica pubblicato che usa chiavi a 56 bit per cifrare blocchi di testo da 64 bit.

Gli input dell'algoritmo di cifratura sono un blocco di testo in chiaro di dimensione  $2w$  bit e una chiave K. Il blocco di testo in chiaro è suddiviso in due metà,  $LE_0$  e  $RE_0$ . Le due metà dei dati attraversano  $n$  round di elaborazione e poi si combinano per produrre il blocco di testo cifrato. Ciascun round  $i$  ha come input  $LE_{i-1}$  e  $RE_{i-1}$  provenienti dal round precedente, e anche una



sottochiave  $K_i$  che deriva dalla K complessiva. In generale, le sottochiavi  $K_i$  differiscono da K e tra di loro.

Per ogni round, viene eseguita una **sostituzione** sulla metà sinistra dei dati, applicando una **funzione di round F** alla metà destra dei dati e considerando poi l'OR-esclusivo dell'output della funzione di round e della metà sinistra dei dati. La funzione di round ha la stessa struttura generale in ogni round, ma è parametrizzata dalla sottochiave di round  $K_i$ . A seguito di questa sostituzione, viene eseguita una permutazione che consiste nello scambio delle due metà dei dati.

La robustezza crittografica di un **cifrario di Feistel** deriva da tre aspetti della sua progettazione:

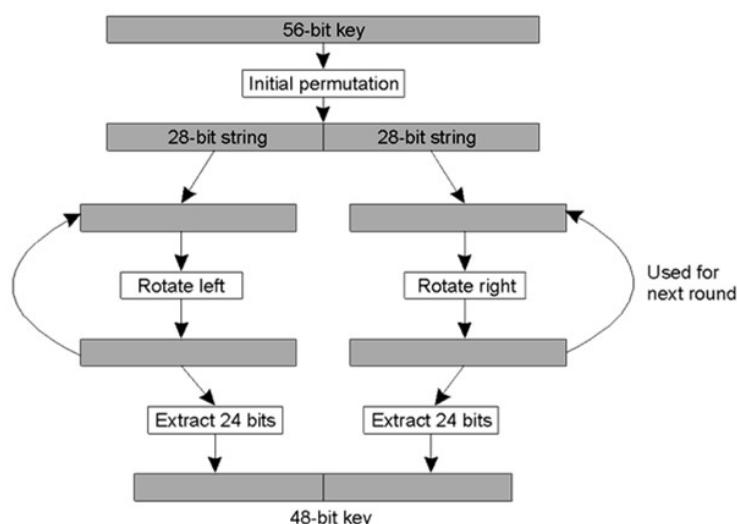
1. **Determinazione del numero di round** Più è grande il numero di round, più è difficile condurre analisi crittografiche, anche per una F relativamente debole. In generale, il criterio di riferimento dovrebbe essere quello di scegliere il numero di round in modo che l'impegno richiesto da un'analisi crittografica sia superiore rispetto a quello richiesto da un semplice attacco a forza bruta per la ricerca della chiave.
2. **Funzione F** Deve risultare difficile “riordinare” la sostituzione eseguita da F in quanto aggiunge  
**confusione** → complicare il più possibile la relazione tra le caratteristiche statistiche del testo cifrato e il valore della chiave di cifratura, per contrastare i tentativi di deduzione della chiave  
Quindi la funzione F deve essere non lineare, più F è non lineare, più sarà difficile qualunque tipo di analisi crittografica.
3. **Generazione delle chiavi** Nel DES, il processo di generazione delle chiavi inizia con una **chiave primaria di 64 bit**. Tuttavia, solo **56 bit effettivi** vengono utilizzati per la cifratura: gli 8 bit rimanenti (uno ogni byte) sono **bit di controllo di parità** e vengono scartati durante il processo. Una volta eliminati questi bit di parità, la chiave di 56 bit subisce una **permutazione iniziale fissa**, chiamata **PC-1 (Permuted Choice 1)**. Questa permutazione riorganizza i bit della chiave e suddivide il risultato in due metà:
  - a. **C<sub>0</sub>**: i primi 28 bit
  - b. **D<sub>0</sub>**: gli ultimi 28 bit

A partire da queste due porzioni, si attiva un processo iterativo chiamato **subkey scheduling**, che serve a generare **16 sottochiavi da 48 bit**, una per ciascuno dei 16 round di cifratura previsti dal DES.

Per ogni round, si eseguono i seguenti passaggi:

- Rotazione a sinistra:** I blocchi C e D vengono entrambi ruotati verso sinistra di **1 o 2 posizioni**, a seconda del round. Il numero di rotazioni è determinato da una **tabella standard** prevista dal protocollo DES.
- Concatenazione:** I nuovi blocchi ottenuti ( $C_n$  e  $D_n$ ) vengono concatenati per formare un blocco da 56 bit.
- Permutazione PC-2 (Permuted Choice 2):** Su questo blocco concatenato viene applicata una seconda permutazione, chiamata **PC-2**, che seleziona **48 bit** e li dispone secondo un ordine specifico. Il risultato è la **sottochiave  $K_n$**  del round corrente.

Questo processo viene ripetuto per tutti i 16 round, producendo così **16 sottochiavi  $K_1, K_2, \dots, K_{16}$** , ciascuna diversa ma derivata dalla chiave iniziale.



Il **Triple DES (3DES)** è un'evoluzione del DES pensata per migliorare la sicurezza, visto che era diventato vulnerabile agli attacchi brute force.

L'idea alla base del 3DES è di applicare il cifrario DES **tre volte di** seguito, con una combinazione di chiavi. Esistono due varianti principali:

### 1. 3DES a due chiavi (2-key 3DES):

- Si cifra con la prima chiave (**K1**).
- Si decifra con la seconda chiave (**K2**).
- Si cifra di nuovo con la prima chiave (**K1**).
- Se qualcuno prova a decifrare il testo cifrato con un solo DES, non otterrà nulla di leggibile.

## 2. 3DES a tre chiavi (3-key 3DES):

- Si cifra con la prima chiave (**K1**).
- Si decifra con la seconda chiave (**K2**).
- Si cifra con la terza chiave (**K3**).

Questa sequenza di **cifratura-decrittazione-cifratura** serve a garantire che, se si usa una chiave specifica, il 3DES sia compatibile con il DES originale. Infatti, se si imposta **K1 = K2 = K3**, il 3DES si comporta come un normale DES, garantendo retrocompatibilità con i vecchi sistemi.

---

## AES

Il National Institute of Standards and Technology ha invitato a presentare proposte per realizzare uno standard per la crittografia, detto **AES (Advanced Encryption Standard)**. Le idee di base avrebbero dovuto essere:

- l'algoritmo doveva essere un cifrario a blocchi simmetrico
- il progetto doveva essere completamente pubblico
- dovevano poter essere impiegate chiavi di 128, 192, e 256 bit
- dovevano essere possibili sia implementazioni software che hardware
- l'algoritmo doveva essere pubblico o adottabile su licenza

Nel 2001 venne scelto **Rijndael**, un algoritmo sviluppato da due crittografi belgi, Vincent Rijmen e Joan Daemen.

Supporta lunghezze di chiavi e dimensioni dei blocchi da 128 a 256 bit, in passi da 32 bit. Questo permetteva di scegliere la lunghezza della chiave e la dimensione dei blocchi in modo indipendente.

Scegliendo chiavi a 128 bit, lo spazio delle chiavi risultò pari a  $2^{128}$  circa  $3 \cdot 10^{38}$  w anche impiegando un bilione di processori, ciascuno in grado di valutare una chiave ogni picosecondo, ci vorrebbero 10<sup>10</sup> anni per ricercare tutto lo spazio delle chiavi. 

Le varie operazioni sono applicate ai byte del blocco di input in diversi round che varia in base alla lunghezza del blocco

Ogni round coinvolge quattro operazioni fondamentali con le quali si crea in una non linearità dei dati e di conseguenza una maggiore sicurezza. Tutte le operazioni sui dati vengono effettuate su di un array bidimensionale, chiamato **State**, composto da un certo numero di righe e colonne che memorizza i byte di input.

Le quattro operazioni effettuate ad ogni round sono:

- **SubBytes**, ogni byte della matrice viene modificato tramite la S-box a 8 bit;
- **ShiftRows**, provvede a spostare le righe della matrice di un parametro, e dipende dal numero di riga. La prima riga resta invariata, la seconda viene spostata di un posto verso sinistra, la terza di due posti e la quarta di tre ecc. In questo modo l'ultima colonna dei dati in ingresso andrà a formare la diagonale della matrice in uscita;
- **MixColumns**, prende i quattro byte di ogni colonna e li combina utilizzando una trasformazione lineare invertibile;
- **AddRoundKey**, combina con uno XOR la chiave di round con la matrice ottenuta dai passaggi precedenti;

Per la fase di decifratura di un blocco cifrato si invertono le operazioni in un numero di round uguale della cifratura.

---

## MODALITÀ OPERATIVE

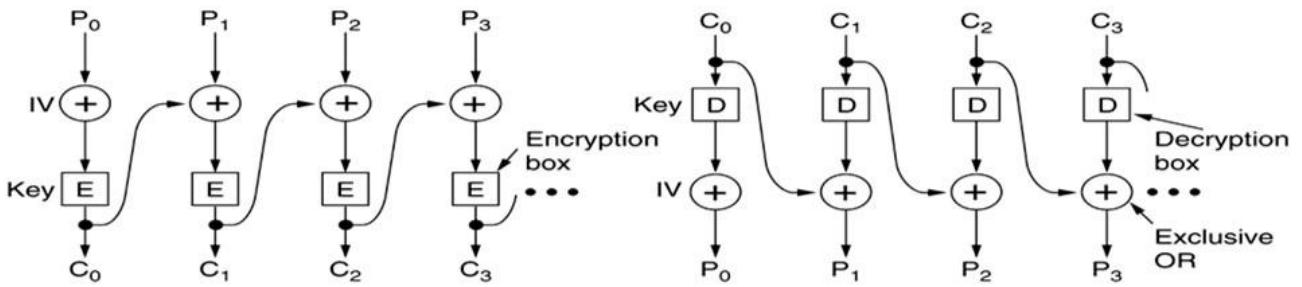
Se abbiamo un file strutturato, diviso in campi, e lo crittografiamo anche il testo cifrato sarà strutturato. Se sapessimo come è strutturato file e conosciamo il testo cifrato sarebbe possibile modificare il contenuto del testo cifrato, modificando le informazioni.

Questo è possibile perché i campi sono fissi e conosciamo la struttura del file. Per evitare questi tipi di attacchi è possibile usare una **concatenazione a blocchi**, un meccanismo in cui ogni blocco di testo in chiaro viene combinato con il blocco cifrato precedente prima di essere cifrato. Quindi dipende da ciò che viene cifrato in precedenza. L'algoritmo si avvia definendo un **Initialization Vector IV** per introdurre casualità.

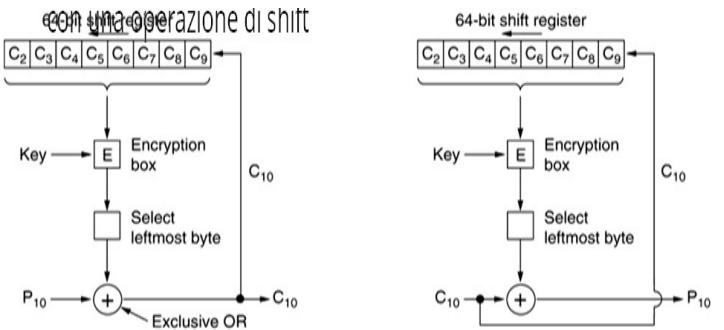
Le principali modalità di concatenazione sono:

- block chaining
- feedback
- stream
- counter

**Block chaining** Prevede che ogni blocco di testo in chiaro venga prima combinato, tramite operazione XOR, con il blocco di testo precedentemente cifrato, e dopo cifrato con la chiave. Il primo blocco viene invece XORato con un valore dell'IV. Questo approccio rende la cifratura **polialfabetica**, cioè anche se la chiave resta invariata, ogni blocco produce un output diverso grazie alla dipendenza dai blocchi precedenti. Un limite è che non consente la cifratura parallela: ogni blocco deve attendere la cifratura di quello che lo precede, rallentando il processo in sistemi ad alta velocità o in contesti di streaming.



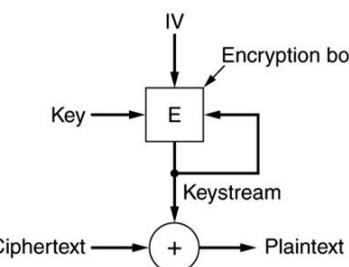
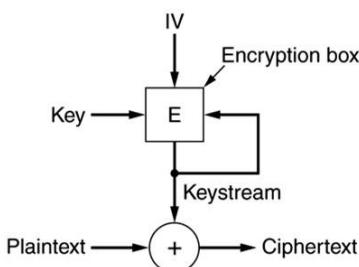
**Feedback** Consente invece di effettuare la cifratura **byte per byte** (o anche bit per bit). In questo schema si utilizza un **registro a scorrimento**, inizializzato con un IV, il cui contenuto viene cifrato a ogni passo. Il primo byte (o bit) dell'output cifrato viene poi XORato con il carattere del testo in chiaro per ottenere il testo cifrato. Questo risultato viene poi reinserito nel registro, che scorre, influenzando così i passaggi successivi. Anche in questo caso si ottiene un comportamento poliafabetico, ma con la flessibilità di operare a livello di singoli caratteri. Tuttavia, un



errore in un singolo bit del testo cifrato può influenzare la decifratura di più byte consecutivi (di solito 8), fino a quando il byte corrotto esce dal registro. Un limite di questo metodo è che se un bit del testo cifrato viene invertito casualmente durante la trasmissione, gli 8 byte che vengono decifrati mentre il byte corrotto si trova nel registro a scorrimento risulteranno anch'essi corrotti, ma non appena il byte corrotto esce dal registro a scorrimento verrà nuovamente generato del testo in chiaro corretto

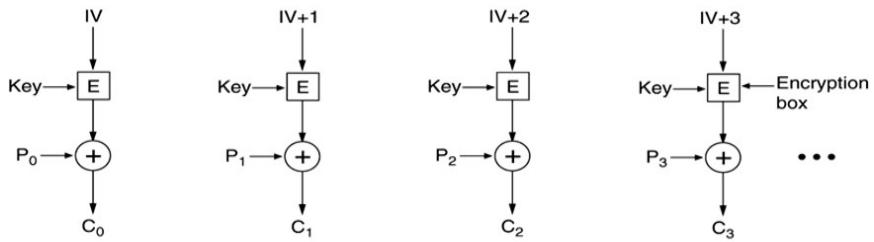
**Stream** Si parte da un IV che viene cifrato con la chiave per produrre un primo blocco di output. Questo blocco viene poi ricifrato per generare il blocco successivo, e così via. Si ottiene così una sequenza pseudo-casuale di blocchi detta **keystream**, che viene poi XORata con il testo in chiaro per ottenere il testo cifrato. La decodifica avviene generando lo stesso keystream e applicando la stessa XOR. Il vantaggio è che un errore di trasmissione non si propaga oltre il bit corrotto, rendendola adatta a canali rumorosi o inaffidabili.

**Counter**  
solo una  
**parallelo**, ma  
**accesso**  
**blocchi**



Permette non  
cifratura **in**  
anche un  
**casuale ai**  
**cifrati**. Il

funzionamento prevede che si parta da un IV iniziale, al quale si aggiunge un contatore incrementale per ogni blocco. Ogni valore del contatore viene cifrato con la chiave per generare un blocco di keystream, da XORato poi con il corrispondente blocco di testo in chiaro. Poiché i valori del contatore possono essere pre-generati, consente un'elevata parallelizzazione ed efficienza. L'unico vincolo è che ogni IV utilizzato con una stessa chiave deve essere **unico**, altrimenti si compromette la sicurezza del sistema.



## SCAMBIO DELLE CHIAVI

La **gestione delle chiavi crittografiche** è il processo di amministrazione o gestione delle chiavi crittografiche per un sistema crittografico. Comprende la generazione, la creazione, la protezione, la conservazione, lo scambio, la sostituzione e l'uso delle chiavi e permette una restrizione selettiva per certe chiavi. Oltre alla restrizione di accesso, la gestione delle chiavi comporta anche il monitoraggio e la registrazione dell'accesso, dell'uso e del contesto di ogni chiave.

La cifratura simmetrica funziona, le due parti coinvolte in uno scambio devono condividere la stessa chiave, e tale chiave deve essere protetta dall'accesso di terzi. È utile cambiare frequentemente la chiave per limitare la quantità di dati che possono essere compresi da un attaccante se dovesse apprendere la chiave. La forza di qualsiasi sistema crittografico risiede nella tecnica di **distribuzione della chiave**, quindi come le parti si scambiano le chiavi per poter comunicare in modo sicuro. È importante che terzi non vengono a scoprire di come la chiave viene scambiata, perché potrebbero capirla e quindi ottenere informazioni.

La distribuzione delle chiavi può essere realizzata in diversi modi, come:

- la chiave può essere consegnata fisicamente;
- Una terza parte può selezionare la chiave e consegnarla fisicamente a chi è coinvolto nella comunicazione
- Se i comunicanti hanno precedentemente e recentemente usato una chiave, una parte può trasmettere la nuova chiave all'altra, cifrata con la vecchia chiave.

- Se i comunicanti hanno ciascuno una connessione cifrata con una terza parte C, C può consegnare una chiave ad A e B attraverso i collegamenti cifrati.

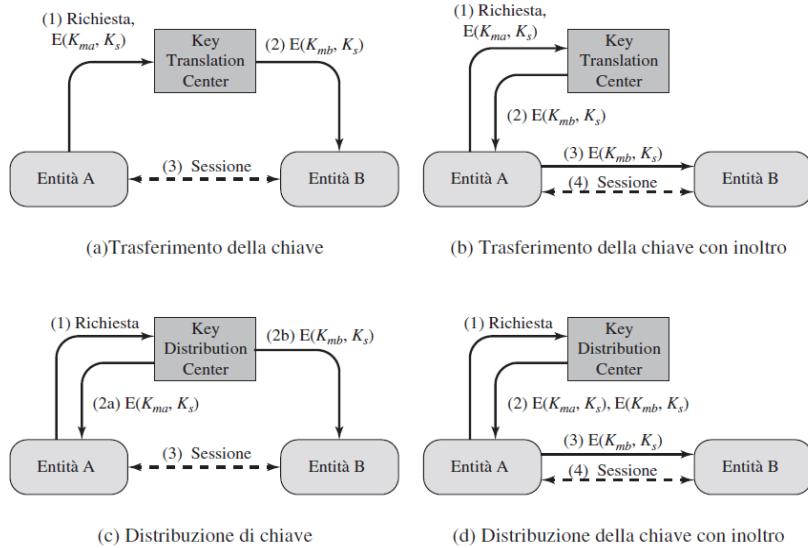
Per la **cifratura end-to-end** su una rete, la consegna manuale è scomoda. In un sistema distribuito, ogni dato utente o server può avere bisogno di impegnarsi in scambi con molti altri utenti e server nel tempo. Così, ogni sistema terminale ha bisogno di un certo numero di chiavi fornite dinamicamente.

Per la crittografia end-to-end un **centro di distribuzione delle chiavi** è responsabile della distribuzione delle chiavi a coppie di utenti (host, processi, applicazioni). Ogni utente deve condividere una chiave unica con il centro di distribuzione delle chiavi ai fini della loro distribuzione.

**Trusted Third Party** Supponiamo di avere una connessione tra le entità A e B, che desiderano scambiare informazioni usando tecniche crittografiche. Hanno bisogno di una **chiave di sessione (Ks)** temporanea che sia valida per la durata di una connessione logica A e B condividono una **chiave master (Kma, Kmb)** di lunga durata con una terza parte che è coinvolta nella distribuzione della chiave di sessione.

Un **Key Translation Center (KTC)** trasferisce chiavi simmetriche per la comunicazione futura tra due entità, delle quali almeno una ha la capacità di generare o acquisire chiavi simmetriche da sola. L'entità A genera o acquisisce una chiave simmetrica da usare come chiave di sessione per la comunicazione con B. A cifra la chiave usando la chiave master che condivide con il KTC e invia la chiave cifrata al KTC. Il KTC decifra la chiave di sessione, cifra nuovamente la chiave di sessione con la chiave master che condivide con B, e invia la chiave di sessione cifrata ad A affinché A la inoltri a B o la invia direttamente a B.

Un **Key Distribution Center (KDC)** genera e distribuisce le chiavi di sessione. L'entità A invia una richiesta al KDC per una chiave simmetrica da usare come chiave di sessione per la comunicazione con B. Il KDC genera una chiave di sessione simmetrica, e poi cifra la chiave di sessione con la chiave master che condivide con A e la invia ad A. Il KDC cifra anche la chiave di sessione con la chiave master che condivide con B e la invia a B. Oppure, invia entrambi i valori di chiave cifrati ad A, e A inoltra a B la chiave di sessione cifrata con la chiave master condivisa dal KDC e da B.

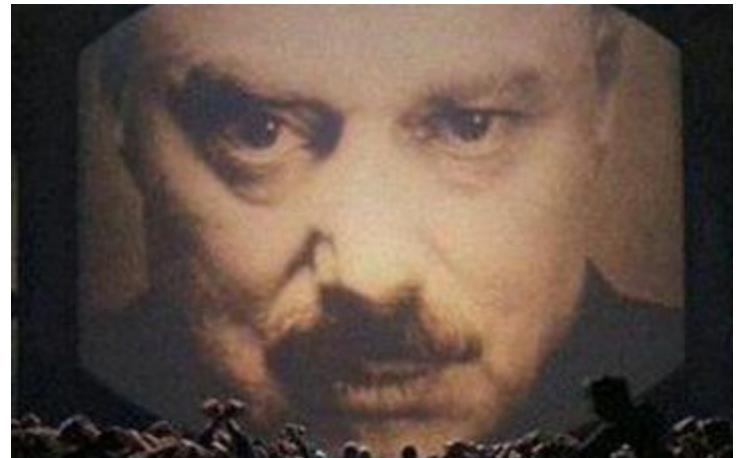


Anche se una TTP può semplificare e rafforzare la fiducia tra le parti, **introduce un punto di fallimento** cioè

- Se la TTP viene compromessa, l'intero sistema può essere a rischio.
- la **fiducia** è un elemento esterno alla tecnologia: si basa su un fornitore di chiavi che spesso è un'azienda e che quindi non è detto che agisca in modo leale e trasparente.

Il fatto che una sola entità possa avere accesso a tutte queste informazioni, se decidesse (o venisse costretta), potrebbe monitorare, controllare o manipolare le comunicazioni di milioni di persone.

Questa **centralizzazione del potere e dell'accesso** è vista con grande sospetto in ambiti governativi, aziendali ad alto rischio ecc.



Il **key agreement** è un processo attraverso cui **due (o più) parti che non si conoscono** possono **negoziare una chiave segreta condivisa**, in modo **sicuro**, anche se stanno comunicando su un canale **non protetto** (come Internet). Questa chiave sarà poi usata per **cifrare e decifrare i messaggi** scambiati tra di loro.

! A differenza della semplice **key transport**, nel key agreement **nessuna delle due parti possiede la chiave dall'inizio**: la costruiscono insieme mediante uno scambio preliminare di informazioni.

Un possibile approccio è quello del **Merkel puzzle** anche se è impossibile da implementare. L'idea alla base è creare un grande numero di "puzzle" crittografici, ognuno dei quali richiede un certo sforzo computazionale per essere risolto.

1. Alice genera un gran numero di "puzzle" (es. 1.000 o 10.000), ognuno con un identificatore, una chiave segreta casuale. Il tutto cifrato con una chiave
2. Alice invia tutti i puzzle a Bob.
3. Bob sceglie un puzzle a caso e lo risolve).
4. Bob scopre l'ID e la chiave segreta del puzzle, e invia l'ID ad Alice.
5. Alice, conoscendo tutti i puzzle, sa a quale chiave segreta corrisponde quell'ID, quindi ora entrambi condividono la stessa chiave.

L'attaccante se vede tutti i **puzzle** (cifrati) e la risposta di Bob, non può sapere quale di questi Bob ha risolto. Dovrebbe provare tutti i puzzle. Anche se l'attaccante è in svantaggio, questo algoritmo risulta oneroso anche per Bob e Alice

- Alice deve creare e cifrare un numero elevato di puzzle
- Bob deve risolverne uno, e anche se è solo uno, è comunque un'operazione che richiede tempo.

Alla base della crittografia moderna c'è una distinzione fondamentale tra due tipi di problemi matematici: **quelli facili** e **quelli difficili**, dal punto di vista computazionale.

- Un **problema "facile"** è un problema per cui esiste un algoritmo efficiente in grado di risolverlo in tempi ragionevoli, anche se i dati in ingresso sono molto grandi.
- Un **problema "difficile"**, invece, è un problema per cui non si conosce alcun algoritmo efficiente. Questi problemi sono spesso facili da enunciare, ma estremamente complessi da risolvere nella pratica.

Alcuni esempi usati in crittografia sono:

- La fattorizzazione di numeri molto grandi (RSA),
- Il logaritmo discreto (Diffie-Hellman),
- Problemi su curve ellittiche, e molti altri.

La cosa interessante è che questi problemi sono spesso facilissimi da generare ma praticamente impossibili da risolvere.

I problemi difficili sono la base teorica della sicurezza crittografica perché permettono di costruire sistemi sicuri, visto che nessuno può risolverli in tempi utili, neanche con enormi risorse computazionali.

**Il protocollo Diffie-Hellman** L'algoritmo di Diffie-Hellman, proposto nel 1976 da Whitfield Diffie e Martin, non cifra messaggi, ma consente a due parti di generare una chiave segreta condivisa attraverso un canale pubblico accordandosi su una chiave segreta comune, anche comunicando su un canale non sicuro, senza che terzi possano ricavare la chiave. La sicurezza del protocollo si basa sulla difficoltà computazionale del calcolo del logaritmo discreto in un campo finito.

Supponiamo che Alice e Bob vogliano scambiarsi una chiave segreta.

1. Alice sceglie un grande **numero primo N** (es. 1024 bit) e un **generatore g** del gruppo moltiplicativo modulo N. Questi due valori possono essere pubblici o forniti da un'autorità fidata.
2. Alice sceglie un **numero casuale segreto**  $a < N$ , calcola  $A = g^a \text{ mod } N$  e invia A a Bob. A questo punto, N, g e A sono pubblici, ma a è noto solo ad Alice.
3. Bob sceglie un numero casuale segreto  $b < N$ , calcola  $B = g^b \text{ mod } N$  e invia B ad Alice. Ora N, g, A e B sono pubblici, ma b è noto solo a Bob.
4. Entrambi calcolano la stessa chiave segreta:
  - a. Alice calcola  $k = B^a \text{ mod } N$
  - b. Bruno calcola  $k = A^b \text{ mod } N$

Poiché:  $(g^b)^a \text{ mod } N = (g^a)^b \text{ mod } N = g^{ab} \text{ mod } N$ . La chiave segreta condivisa k risulta identica per entrambi.

### Esempio

Per chiarire il funzionamento, ecco un esempio con numeri piccoli:

- Alice sceglie  $N = 17$ ,  $g = 3$  (3 è un generatore modulo 17)
- Alice sceglie  $a = 6$  e calcola  $A = 3^6 \text{ mod } 17 = 15$
- Bruno sceglie  $b = 11$  e calcola  $B = 3^{11} \text{ mod } 17 = 7$

Scambio:

- Alice riceve B = 7 e calcola  $k = 7^6 \text{ mod } 17 = 9$
- Bruno riceve A = 15 e calcola  $k = 15^{11} \text{ mod } 17 = 9$

La chiave segreta condivisa è quindi  $k = 9$ .

Chiunque intercetti i valori N, g, A, B non può ricavare facilmente la chiave  
 $k=g^{ab} \text{ mod } N$

senza conoscere a o b. In teoria, potrebbe farlo calcolando il logaritmo discreto ( $a = \log_g A$  o  $b = \log_g B$ ), ma questo calcolo è computazionalmente impossibile per numeri sufficientemente grandi (tipicamente  $N \geq 2048$  bit).

Il protocollo Diffie-Hellman, da solo, non garantisce l'autenticazione. È quindi vulnerabile a un attacco MITM:

- Un terzo, ad esempio Carlo, intercetta A, genera un proprio valore B' e lo invia a Bruno fingendosi Alice.
- Poi intercetta il valore B di Bruno, genera un proprio A' e lo invia ad Alice fingendosi Bruno.

A questo punto:

- Alice condivide la chiave con Carlo, credendo sia Bruno.
- Bruno condivide una diversa chiave con Carlo, credendo sia Alice.
- Carlo può decifrare e ritrasmettere i messaggi, leggendo tutto senza che Alice e Bruno se ne accorgano.

Per proteggere lo scambio contro attacchi MITM, si può affiancare al protocollo un meccanismo di autenticazione basato su certificati digitali. Alice e Bruno, ad esempio, possono autenticarsi reciprocamente firmando digitalmente i valori inviati, utilizzando le proprie chiavi private. I certificati che contengono le chiavi pubbliche sono rilasciati da un ente certificatore (**CA – Certification Authority**) fidato.

---

## CRITTOGRAFIA A CHIAVE PUBBLICA

---

La crittografia a chiave pubblica, o asimmetrica, è un sistema crittografico in cui ogni utente dispone di una coppia di chiavi:

- **chiave pubblica (e)**: può essere condivisa con chiunque.
- **chiave privata (d)**: deve rimanere segreta e conosciuta solo dal proprietario.

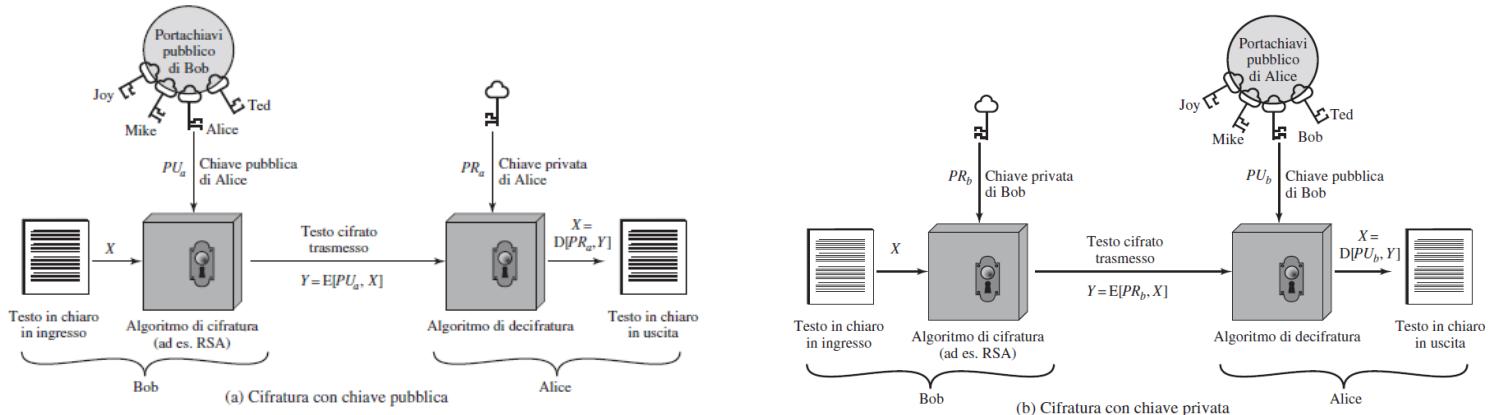
Se una viene usata per la cifratura, l'altra è usata per la decifratura.

Le due chiavi sono collegate tra loro da funzioni matematiche piuttosto che su sostituzione e permutazione il che rende il sistema sicuro perché anche conoscendo la chiave pubblica, è computazionalmente impossibile risalire alla chiave privata.

## FUNZIONAMENTO

1. ogni utente genera una coppia di chiavi per usarle per la cifratura e la decifratura dei messaggi;
2. ogni utente colloca una delle due chiavi in un registro pubblico o su altri archivi accessibili. Questa è la chiave pubblica. L'altra chiave è mantenuta privata. Ogni utente mantiene una collezione di chiavi pubbliche ottenute da altri;
3. Se Bob desidera mandare un messaggio confidenziale ad Alice, Bob cifra il messaggio usando la chiave pubblica di Alice;

4. quando Alice riceve il messaggio, lo decifra usando la propria chiave privata. Nessun altro può decifrare il messaggio perché soltanto Alice è a conoscenza della propria chiave privata.



Tutti i partecipanti hanno accesso alle chiavi pubbliche, mentre le chiavi private sono generate in proprio da ogni partecipante e di quindi non è mai richiesta la distribuzione. Finché la chiave privata di un utente rimane protetta e segreta, le comunicazioni in ingresso sono sicure. In qualunque momento, un sistema può cambiare la propria chiave privata e pubblicare la corrispondente chiave pubblica per rimpiazzare quella vecchia.

Questo schema **NON** fornisce la **confidenzialità**, il messaggio inviato è al sicuro dalle modifiche, ma non dalle intercettazioni. Questo è evidente nel caso di una firma basata su una parte del messaggio, poiché il resto del messaggio è trasmesso in chiaro.

Anche nel caso di una cifratura completa non c'è alcuna protezione della confidenzialità, perché qualunque osservatore può decifrare il messaggio adoperando la chiave pubblica del mittente.

- A prepara un messaggio da inviare a B e lo cifra usando la propria chiave privata prima di trasmetterlo.
- B può decifrare il messaggio usando la chiave pubblica di A.

Poiché il messaggio è stato cifrato usando la chiave privata di A, soltanto A può avere preparato il messaggio, è impossibile modificare il messaggio senza avere accesso alla chiave privata di A, quindi il messaggio è autenticato sia in termini di sorgente che in termini di integrità dei dati.

L'intero messaggio è cifrato, il che pur validando sia autore che contenuto, richiede una grande quantità di spazio di archiviazione. Ogni documento deve essere mantenuto in chiaro per essere usato e testo cifrato così che l'origine e il contenuto possano essere verificati in caso di disputa.

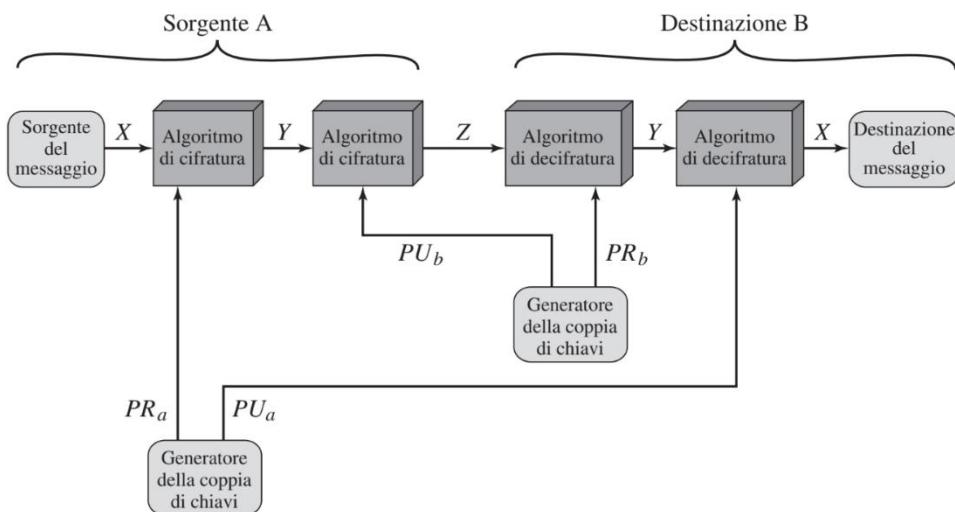
Una maniera più efficiente di ottenere lo stesso risultato consiste nel cifrare un blocco ridotto di bit, che sia funzione del documento. Questo blocco è chiamato

**autenticatore** e deve avere la proprietà che non sia possibile cambiare il documento senza cambiare l'autenticatore.

Se l'autenticatore è cifrato con la chiave privata del mittente, esso serve da firma che verifica l'origine, il contenuto e l'ordine di sequenza.

Questo però non garantisce la confidenzialità, quindi può essere intercettato perché uno che intercetta il messaggio lo può decifrare usando la chiave pubblica del mittente.

È però possibile fornire sia la funzione di autenticazione che la confidenzialità tramite un doppio utilizzo dello schema a chiave pubblica; si comincia cifrando un messaggio, usando la chiave privata del mittente. Ciò fornisce la firma digitale. Quindi, si cifra nuovamente usando al chiave pubblica del destinatario. Il testo cifrato finale può essere decifrato solo dal legittimo destinatario, che è l'unico a possedere la chiave privata corrispondente. Quindi è ottenuta la confidenzialità. Lo svantaggio di questo approccio è che l'algoritmo a chiave pubblica, che è di suo complesso, deve essere eseguito quattro volte invece di due per ogni comunicazione.



I principali algoritmi di crittografia asimmetrica includono:

- **RSA**: basato sulla difficoltà di fattorizzare numeri primi molto grandi.
- **ElGamal**: si basa sul problema del logaritmo discreto.
- **DSA (Digital Signature Algorithm)**: pensato per le firme digitali.
- **ECC (Elliptic Curve Cryptography)**: offre lo stesso livello di sicurezza con chiavi più corte.

## RSA Rivest-Shamir-Adleman

L'RSA è un cifrario nel quale il testo in chiaro e il testo cifrato sono interi tra 0 e  $n-1$  per un qualche  $n$ . Una dimensione tipica per  $n$  è di 1024 bit, ovvero 309 cifre decimali. Quindi  $n$  è inferiore a  $2^{1024}$ .

L'RSA fa uso di una espressione con esponenziali. Il testo in chiaro è cifrato in blocchi, e ogni blocco ha un valore binario minore di un qualche numero  $n$ .

Ovvero, la dimensione del blocco deve essere minore o uguale a  $\log_2(n) + 1$ ; in pratica la dimensione del blocco è di  $i$  bit, con  $2i < n \leq 2i+1$ .

### Cifratura e decifratura

$$\text{Cifratura} \rightarrow C = M^e \pmod{n}$$

$$\text{Decifratura} \rightarrow M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}$$

### Chiavi e Sicurezza

La **chiave pubblica**  $PU=\{e,n\}$  è utilizzata dal mittente per cifrare il messaggio.

La **chiave privata**  $PR=\{d,n\}$  è utilizzata dal destinatario per decifrare il messaggio.

Sia mittente che destinatario devono conoscere il valore di  $n$ . Il mittente conosce il valore di  $e$ , e solo il destinatario conosce il valore di  $d$ .

### Validazione dell'algoritmo

1. Deve essere possibile trovare dei valori di  $e$ ,  $d$  e  $n$  tali che  $M^{ed} \pmod{n} = M$  per qualsiasi  $M < n$ .

#### Dim.

La relazione è vera se  $e$  e  $d$  sono inversi moltiplicativi modulo  $\phi(n)$ , dove  $\phi(n)$  è la funzione toziente di Eulero. Per  $p, q$  primi,  $\phi(pq)=(p-1)(q-1)$ .

La relazione tra  $e$  e  $d$  può essere espressa come

$$ed \pmod{\phi(n)} = 1$$

Ciò è equivalente a dire

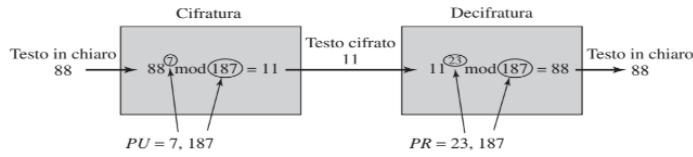
$$ed \equiv 1 \pmod{\phi(n)}$$

$$d \equiv e^{-1} \pmod{\phi(n)}$$

Cioè,  $e$  e  $d$  sono inversi moltiplicativi modulo  $\phi(n)$ . Secondo le regole dell'aritmetica modulare, ciò è vero solo se  $d$  (e di conseguenza  $e$ ) è primo relativo con  $\phi(n)$ . Equivalentemente  $\text{mcd}(\phi(n), d) = 1$ .

2. Deve essere relativamente semplice calcolare  $M^e \pmod{n}$  e  $C^d \pmod{n}$  per qualsiasi  $M < n$ .
3. Non sia fattibile determinare  $d$  dati  $e$  ed  $n$

| Generazione delle chiavi di Alice                              |   |
|--|---|
| Selezionare $p, q$   | $p \neq q$ entrambi primi, $p \neq q$         |
| Calcolare $n = p \times q$                                     |   |
| Calcolare $\phi(n) = (p-1)(q-1)$                               |   |
| Selezionare un intero $e$                                      | $\text{mcd}(\phi(n), e) = 1; 1 < e < \phi(n)$ |
| Calcolare $d$  | $d \equiv e^{-1} \pmod{\phi(n)}$              |
| Chiave pubblica  | $PU = \{e, n\}$                               |
| Chiave privata   | $PR = \{d, n\}$                               |
| Cifratura effettuata da Bob con la chiave pubblica di Alice    |   |
| Testo in chiaro:   | $M < n$                                       |
| Testo cifrato:   | $C = M^e \pmod{n}$                            |
| Decifratura effettuata da Alice con la chiave privata di Alice |   |
| Testo cifrato:   | $C$   |
| Testo in chiaro:   | $M = C^d \pmod{n}$                            |



La crittografia asimmetrica può essere impiegata per creare cripto-sistemi più complessi che permettono di garantire alcuni principi della CIA.

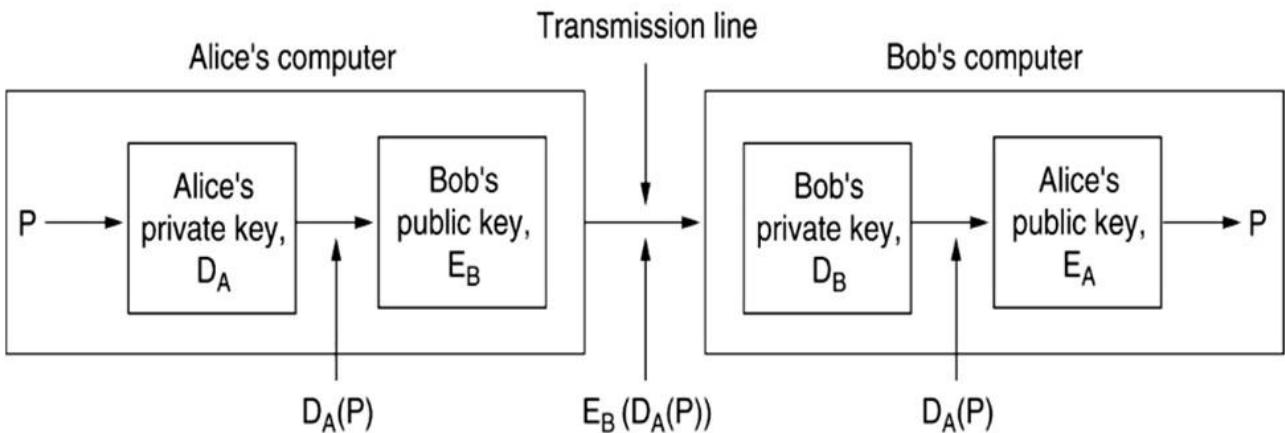
## FIRMA DIGITALE

La **firma digitale** è un metodo matematico per dimostrare l'autenticità di un documento digitale autenticando l'autore di un documento.

Una firma digitale è un valore calcolato con un algoritmo crittografico e associato a dei dati in modo tale che qualsiasi destinatario possa utilizzarla per **verificare l'origine e l'integrità dei dati**. In genere, il firmatario utilizza la propria **chiave privata per generare la firma**, e chiunque sia in possesso della **chiave pubblica corrispondente può verificarne la validità**. Questa fornisce un insieme di caratteristiche di sicurezza difficili da implementare in altro modo.

### Proprietà di una firma digitale:

- Deve verificare l'autore, la data e l'orario della firma.
- Deve autenticare i contenuti al momento della firma.
- Deve essere verificabile da terze parti per risolvere dispute.
- Deve essere un pattern di bit che dipende dal messaggio da firmare.
- Deve utilizzare informazioni conosciute solo dal mittente per prevenire falsificazione e ripudio.
- Deve essere relativamente semplice da produrre, riconoscere e verificare.
- Deve essere computazionalmente infattibile falsificare.
- Deve essere agevole mantenerne una copia in memoria.



- Alice usa la sua chiave privata  $D_A$  per calcolare  $D_A(P)$  e poi cifra ciò che ha ottenuto con la chiave pubblica del destinatario Bob, ottenendo  $E_B(D_A(P))$
- Bob, alla ricezione, effettua il percorso inverso, applicando prima la sua chiave privata e poi quella pubblica del sender Alice, cioè  $D_B(E_A(P))$

Da questo si capisce che il mittente deve esser per forza Alice perché, ha usato la sua chiave privata e che il destinatario era Bob perché ha decifrato con la sua chiave pubblica.

- sono assicurate segretezza, autenticazione del mittente e del destinatario e il non ripudio
- non è assicurata l'integrità, perché terzi possono intromettersi nella comunicazione

I dati devono rimanere invariati se qualcuno gli altera cambia anche la firma. Quindi quando il destinatario riceve il messaggio la confronta con la sua firma, questa non corrisponde. L'attaccante potrebbe pensare di rigenerare la chiave, ma questo è difficile computazionalmente,

Quindi l'importante è rendere segreta la firma, il messaggio può essere anche in chiaro. DELLA PRIVACY CE NE FOTTIAMO.

Per garantire l'autenticazione del messaggio utilizziamo un **algoritmo di hash**, una **funzione hash H** che accetta in input un blocco di dati  $M$  di lunghezza variabile e produce un risultato di dimensione fissa  $h = H(M)$ , indicato come **codice hash**. Un cambiamento anche di un solo bit di  $M$  porta ad un grande cambiamento del valore hash.

### Proprietà

1. La funzione di hash deve essere impossibile da invertire
2. Si devono ridurre al minimo le collisioni in modo che due messaggi diversi abbiano due hash diversi



3. Un cambiamento anche di un solo bit di M porta ad un grande cambiamento del valore hash

## Algoritmi di hash

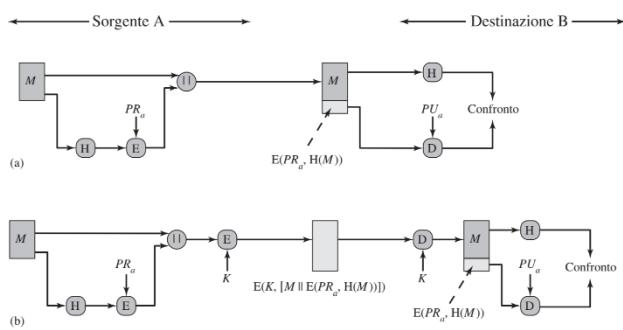
- MD5
- SHA-1
- SHA-2

Quando una funzione hash viene utilizzata per fornire l'autenticazione del messaggio, il valore della funzione hash è spesso indicato come **digest** del messaggio.

✖ Il mittente calcola un valore hash in funzione dei bit del messaggio e trasmette sia il valore hash che il messaggio. Il destinatario esegue lo stesso calcolo dell'hash sui bit del messaggio e confronta questo valore con il valore hash ricevuto. Se non c'è corrispondenza, il destinatario sa che il messaggio è stato alterato.

Il digest è mandato in chiaro, quindi chi si intromette, potrebbe modificare il digest e il mittente non si accorgerebbe della modifica.

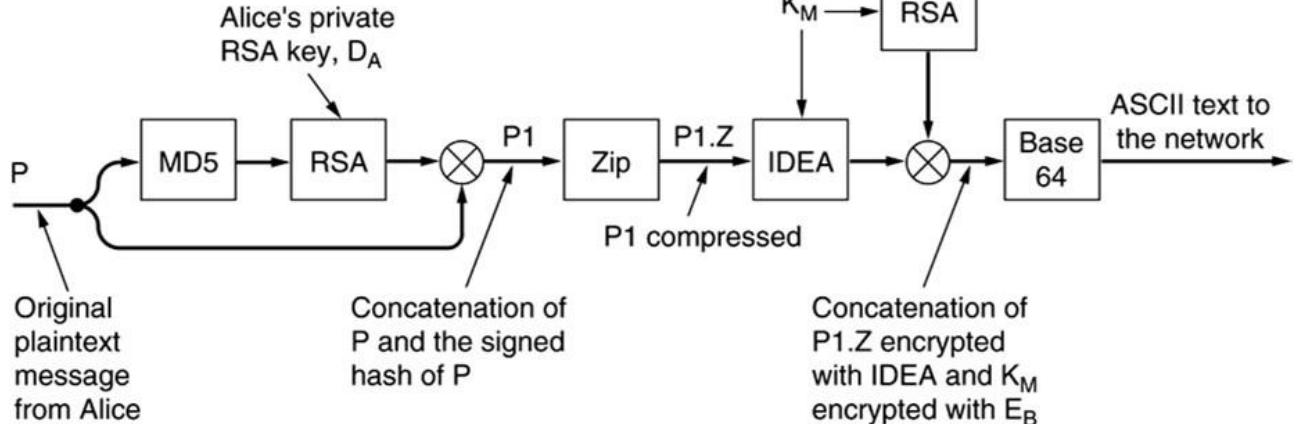
- a. Il codice hash è cifrato, usando la cifratura a chiave pubblica, con la chiave privata del mittente. Fornisce anche una firma digitale, perché solo il mittente potrebbe aver prodotto il codice hash cifrato.
- b. Se si vuole ottenere sia la confidenzialità che la firma digitale, allora il messaggio più il codice hash cifrato con la chiave privata possono essere cifrati usando una chiave simmetrica segreta



**Pretty Good Privacy** Pretty Good Privacy (PGP) è un sistema di crittografia e firma digitale che garantisce la riservatezza, l'integrità e l'autenticità per proteggere le comunicazioni via email.

$K_M$  : One-time message key for IDEA

$\otimes$  : Concatenation



PGP combina sia **crittografia simmetrica** che **crittografia asimmetrica** (a chiave pubblica e privata).

## PEC

La **PEC** (Posta Elettronica Certificata) è un sistema di posta elettronica che garantisce la **valida certificazione legale** dei messaggi scambiati tra mittente e destinatario. Ha lo stesso valore legale di una raccomandata con avviso di ricevimento, grazie alla sua capacità di certificare:

1. L'invio del messaggio.
2. La ricezione del messaggio da parte del destinatario.
3. L'integrità del contenuto del messaggio, cioè che non è stato alterato durante il trasferimento

## GESTIONE DELLE CHIAVI

Il problema della crittografia a chiave pubblica è che per poter comunicare due utenti devono conoscere le rispettive chiavi pubbliche che possono essere trovate in pubblici registri ecc... ma il problema è che non abbiamo la certezza che quella chiave sia di quella persona. (MIM)

Alcuni modi di garantire ciò è che la chiave può essere distribuita da società trusted (KDC) che distribuiscono le chiavi private e pubbliche degli utenti ma una soluzione migliore.

## Certification Authority

Le **Certification of Authority** consistono in una chiave pubblica, un identificatore del proprietario della chiave, e con l'intero blocco firmato da una terza parte fidata. La terza parte è un'autorità di certificazione, come un'agenzia

governativa o un'istituzione finanziaria, di cui la comunità degli utenti si fida. Un utente può presentare la sua chiave pubblica all'autorità in modo sicuro e ottenere un certificato. L'utente fa una richiesta di un certificato all'authority

1. che genera una coppia di chiavi, una pubblica e una privata
2. che associa poi all'anagrafica della persona
3. e la certifica con la propria chiave pubblica
4. L'utente può poi pubblicare il certificato
- 5.

Chiunque abbia bisogno della chiave pubblica di questo utente può ottenere il certificato e verificarne la validità attraverso la firma pubblica dell'authority allegato in modo da verificare che il certificato sia stato creato dall'autorità.

In questo modo anche se qualcuno dovesse intercettare il certificato e corromperlo; quando l'altra parte verificherà l'autenticità del certificato noterà subito che non è quello originale perché, quando calcola l'hash SHA-1 del certificato e lo confronta con il valore ottenuto applicando la chiave pubblica di CA alla firma del certificato, vedrà che sono diversi.

## REQUISITI

1. Qualsiasi partecipante deve poter leggere un certificato per determinare il nome e la chiave pubblica del proprietario del certificato.
2. Qualsiasi partecipante deve poter verificare che il certificato provenga dall'autorità di certificazione e non sia contraffatto.
3. Solo l'autorità di certificazione deve poter creare e aggiornare i certificati.
4. Qualsiasi partecipante deve poter verificare la validità temporale del certificato.

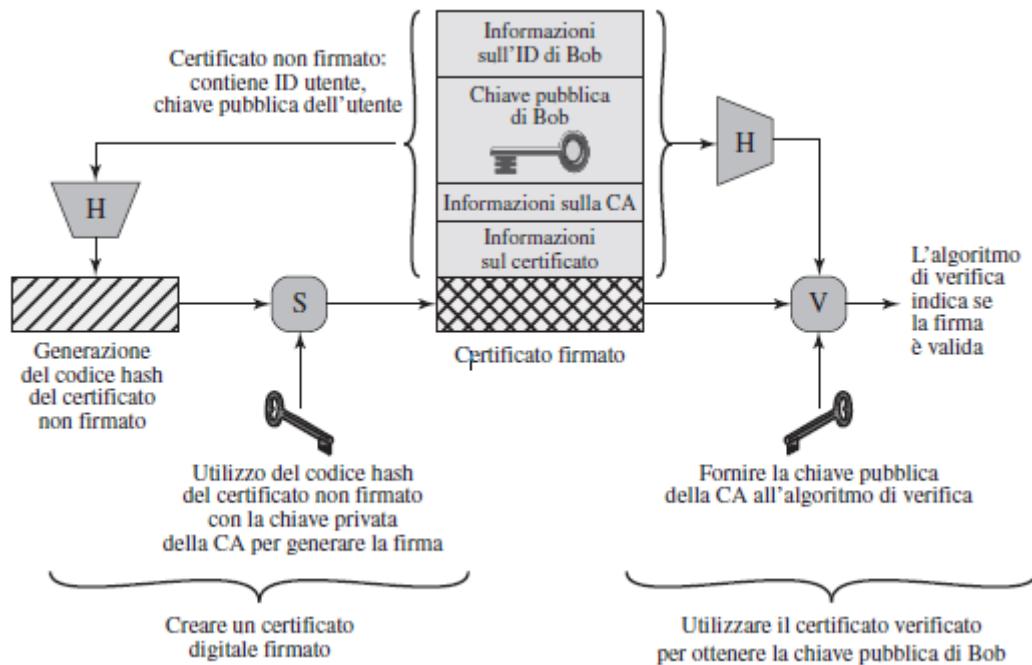
Ogni certificato include un periodo di validità. Di solito, un nuovo certificato viene emesso appena prima della scadenza di quello vecchio. A volte si può revocare un certificato prima che scada, per vari motivi:

1. la chiave privata dell'utente sia compromessa.
2. l'utente non è più certificato da questa CA. Le ragioni di ciò includono che il nome del soggetto è cambiato, il certificato è stato sostituito, o il certificato non è stato emesso in conformità con le politiche della CA.
3. certificato della CA sia compromesso.

Uno standard, proposto da Kohnfelder, è quello di **ITU-T X.509** che si basa sull'uso della crittografia a chiave pubblica e delle firme digitali.

Il certificato per la chiave pubblica di Bob include informazioni uniche di identificazione per Bob, la chiave pubblica di Bob, e informazioni di identificazione sulla autorità di certificazione più info sul certificato.

Queste informazioni vengono poi firmate calcolando un valore hash delle informazioni (SHA-1) e generando una firma digitale usando il valore hash e la chiave privata della CA. Bob può quindi trasmettere questo certificato ad altri utenti, o allegare il certificato a qualsiasi documento o blocco di dati che firma. Chiunque abbia bisogno di usare la chiave pubblica di Bob può essere sicuro che la chiave pubblica contenuta nel certificato di Bob sia valida perché il certificato è firmato dalla CA fidata.



## FORMATO CA

5. **Versione** Differenzia le versioni successive del formato del certificato; il valore predefinito è la versione 1. Se sono presenti l'identificatore unico dell'emittente l'identificatore unico del soggetto, il valore deve essere la versione 2. Se sono presenti una o più estensioni, la versione deve essere la versione 3.
6. **Numero seriale** Un valore intero unico all'interno della CA emittente che è inequivocabilmente associato a questo certificato.
7. **Identificatore dell'algoritmo di firma** L'algoritmo usato per firmare il certificato insieme a qualsiasi parametro associato.
8. **Nome dell'emittente Nome del soggetto** Il nome dell'utente a cui si riferisce questo certificato. Cioè, questo certificato certifica la chiave pubblica del soggetto che possiede la chiave privata corrispondente.
9. **Informazioni sulla chiave pubblica del soggetto** La chiave pubblica del soggetto, più un identificatore dell'algoritmo per il quale questa chiave deve essere usata, insieme a qualsiasi parametro associato.

10. **Identificatore unico dell'emittente** Un campo stringa di bit opzionale usato per identificare in modo univoco la CA emittente nel caso in cui il nome X.500 sia stato riutilizzato per entità diverse.
11. **Identificatore unico del soggetto** – Un campo stringa opzionale di bit usato per identificare in modo univoco il soggetto nel caso in cui il nome X.500 sia stato riutilizzato per entità diverse.
12. **Estensioni** Un insieme di uno o più campi di estensione. Le estensioni sono state aggiunte nella versione 3
13. **Firma** Copre tutti gli altri campi del certificato. Una componente di questo campo è la firma digitale applicata agli altri campi del certificato. Questo campo include l'identificatore dell'algoritmo di firma.
14. **Lista CA revocati** Ogni CA deve mantenere una lista di tutti i certificati revocati ma non scaduti emessi da quella CA, compresi, sia quelli rilasciati agli utenti, che quelli rilasciati ad altre CA. Ogni lista di revoca dei certificati pubblicata nella directory è firmata dall'emittente e include il nome dell'emittente, la data in cui la lista è stata creata, la data in cui è prevista l'emissione della prossima CRL e una voce per ogni certificato revocato. Ogni voce consiste nel numero di serie di un certificato e nella data di revoca per quel certificato. Poiché i numeri di serie sono unici all'interno di una CA, il numero di serie è sufficiente per identificare il certificato.

Affidarsi a una sola CA per emettere tutti i certificati sarebbe un *point of failure* per via del carico di lavoro e della sicurezza della chiave che potrebbe essere facilmente rubata visto che tutti i server avrebbe la stessa chiave privata. Una soluzione consiste nell'avere diverse CA, tutte gestite dalla stessa organizzazione e tutte con la stessa chiave privata usata per firmare i certificati.

## Public-Key Infrastructure PKI

Le **PKI** sono un insieme di politiche, processi, piattaforme server, software e workstation usate allo scopo di amministrare i certificati e le coppie di chiavi pubbliche e private, inclusa la capacità di emettere, mantenere revocare i certificati a chiave pubblica. L'obiettivo principale per lo sviluppo di una PKI è quello di consentire un'acquisizione sicura, conveniente ed efficiente delle chiavi pubbliche.

### Architettura

**Autorità di registrazione (RA- root)** delega molte delle funzioni amministrative svolte normalmente da una CA. La RA è normalmente associata al processo di registrazione dell'entità terminale. Questo include la verifica dell'identità dell'entità terminale che tenta di registrarsi con la PKI e ottenere un certificato

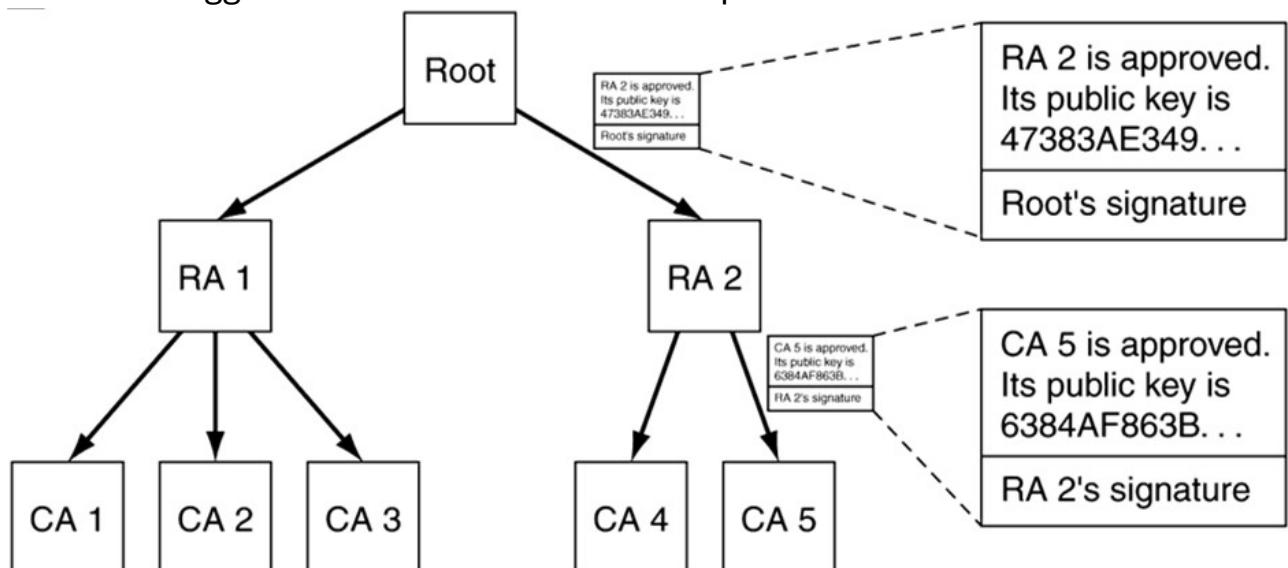
per la sua chiave pubblica.

**Autorità di certificazione (CA)** Un'autorità di cui si fidano uno o più utenti per creare e assegnare certificati a chiave pubblica. L'autorità di certificazione può creare le chiavi dei soggetti e firmare digitalmente i certificati a chiave pubblica, il che lega e effettivamente il nome del soggetto alla chiave pubblica. Le CA sono anche responsabili dell'emissione di liste di revoca dei certificati (CRL). La CRL identifica i certificati precedentemente emessi dalla CA che vengono revocati prima della loro data di termine di validità. Un certificato potrebbe essere revocato perché si presume che la chiave privata dell'utente sia compromessa, l'utente non è più certificato da questa CA o si presume che il certificato sia compromesso.

**Repository** metodo usato per immagazzinare e recuperare informazioni relative alla PKI. Può anche essere qualcosa di semplice, come un mezzo per recuperare un semplice file su un server remoto tramite il File Transfer Protocol (FTP) o l'Hyper Text Transfer Protocol (HTTP).

**Relying Party (RP)** Qualsiasi utente o agente che si basa sui dati in un certificato per prendere decisioni.

**Entità terminale** cioè l'utente finale (un dispositivo, come un router o un server, un processo o qualsiasi elemento) che può essere identificato nel nome del soggetto di un certificato di chiave pubblica.



In una **Public Key Infrastructure (PKI)**, quando un'entità vuole ottenere la chiave pubblica di un'altra per comunicare in modo sicuro, non si fida

cieicamente della chiave ricevuta. È necessario **verificare l'autenticità** della chiave pubblica attraverso una catena di certificati digitali firmati.

Ogni certificato è **firmato digitalmente** da un'autorità di certificazione (CA) superiore nella gerarchia. Tuttavia, non sempre l'entità che riceve il certificato conosce direttamente l'autorità che lo ha firmato. In questi casi, è necessario **risalire una catena di certificati**, in cui ogni autorità giustifica la propria legittimità presentando un certificato firmato da un'autorità ancora superiore. Questo processo continua finché non si arriva a un'autorità considerata **universalmente affidabile**, nota come **root CA**. Le chiavi pubbliche delle root CA sono **preinstallate nei sistemi o nei browser** e sono quindi assunte come affidabili a priori. Queste sono chiamate anche **trust anchors** (ancore fiduciari). Per facilitare il processo, l'entità che fornisce il proprio certificato può anche includere tutti i certificati intermedi necessari per costruire la catena di fiducia, rendendo più semplice la verifica per l'altra parte.

L'intero meccanismo è reso sicuro dal fatto che:

- Ogni certificato è firmato digitalmente, quindi qualsiasi modifica è facilmente rilevabile.
- La fiducia si propaga lungo una **catena di certificati firmati** fino a una root conosciuta e fidata.

Questo approccio è chiamato **chain of trust** o **certification path**, ed è **largamente utilizzato** in sistemi crittografici reali, come nel caso delle connessioni HTTPS via browser.

## AUTENTICAZIONE

L'autenticazione degli utenti consiste nell'identificare l'utente presso il sistema presentando una credenziale, come l'ID utente; il sistema verifica l'utente attraverso lo scambio di informazioni di autenticazione.

L'autenticazione dell'utente è il modo per stabilire la validità della identità presunta.

I protocolli di autenticazione si basano su crittografia a chiave segreta (simmetrica) o a chiave pubblica. Il loro funzionamento permette:

- di stabilire l'identità delle parti per la soddisfazione di entrambi
- poi le parti possono iniziare la comunicazione effettiva o garantire l'accesso ad una risorsa

Durante l'esecuzione del protocollo, vengono scambiati diversi messaggi tra le entità coinvolte. che attraversando canali non sicuri sono soggetti a intercettazione, modifica o ritrasmissione da parte di un attaccante con l'obiettivo di ingannare le parti o interferire con la comunicazione. Nonostante queste possibili minacce ogni parte può **autenticare l'identità dell'altra** e quindi di sicura di comunicare effettivamente con il soggetto desiderato (non con un impostore).

Viene generata e condivisa in modo sicuro una chiave segreta, chiamata **chiave di sessione**, che le due parti utilizzeranno per **proteggere i messaggi futuri** attraverso tecniche di crittografia simmetrica.

Questo tipo di protocollo è alla base di molte **comunicazioni sicure su reti non fidate**, come Internet, e viene utilizzato in applicazioni come login, scambi di dati sensibili e protocolli crittografici complessi.

Si usano chiavi diverse, scelta casualmente, per ciascuna nuova connessione per minimizzare la quantità di traffico inviato usando le chiavi permanenti degli utenti (secrete oppure pubbliche), ridurre la quantità di testo cifrato che può cadere nelle mani di un intruso e minimizzare i rischi connessi alla situazione in cui un processo va in crash e il relativo *core dump* cade nelle mani sbagliate.

Per aggiungere casualità gli protocolli di autenticazione prevedono di lanciare una **sfida** fra le parti che desiderano comunicare. La parte che avvia la comunicazione invia, un numero casuale all'altra parte, la quale provvede a trasformarlo tramite una regola ed a restituirlo.

Il numero casuale impiegato è detto **nounce** e ciò rende molto più difficile gli attacchi.

**CHALLENGE ACCEPTED**



L'**AP in chiaro** consiste che una parte manda all'altra un messaggio dicendo chi è, quindi l'autenticazione avviene tramite scambio di username. Questo protocollo ha il problema che il ricevente non ha la certezza che chi manda il messaggio sia effettivamente la persona con la quale voleva comunicare, perché essendo il messaggio in chiaro può essere facilmente intercettato. Una soluzione potrebbe essere un **AP basato su IP**, dove viene mandato username più IP per autenticazione. Anche in questo caso il ricevente non può sapere con certezza con chi sta attaccando perché l'intruso con un attacco di IP spoofing può inviare pacchetti con **indirizzi IP falsificati**. In questo modo, l'attaccante può fingersi un utente legittimo, inviando un pacchetto che apparentemente proviene da un IP autorizzato, pur non possedendolo realmente.

Potremmo usare un **AP basato su password in chiaro**, in cui viene mandato username e password. Anche questo non funziona, perché essendo che vengono mandate in chiaro quindi con un **attacco di playback** si possono intercettare i messaggi validi scambiati tra due entità durante una comunicazione, e poi riprodurli in un secondo momento per ingannare il destinatario, facendogli credere che il messaggio sia nuovo, valido e ancora proveniente dal mittente originale. Anche cifrandole in problema è lo stesso.

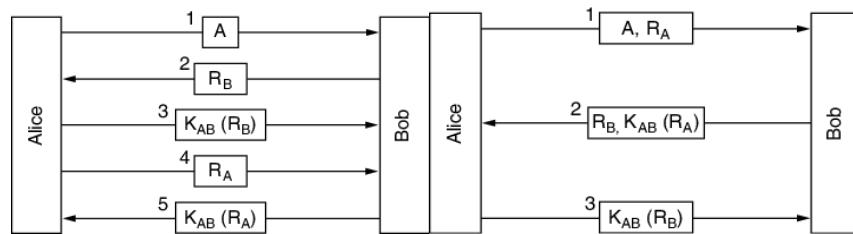
Non funziona niente 😐

Possiamo provare con **AP a due vie con sfida**. Supponiamo Alice e Bob abbiano già una chiave segreta nota a entrambi,  $K_{AB}$ .

1. Nel messaggio 1, Alice invia a Bob la sua identità  $A$  usando una modalità che Bob è in grado di capire. Bob non può sapere se il messaggio viene da Alice o da Trudy.
2. Crea una richiesta per Alice nella forma di un numero casuale grande,  $R_B$ , che invia in chiaro nel messaggio 2.
3. Alice cifra il messaggio usando della chiave condivisa con Bob e inviando il messaggio 3 con il testo cifrato  $K_{AB}(R_B)$ . Quando Bob riceve il messaggio sa immediatamente che proveniva da Alice, in quanto Trudy non conosce  $K_{AB}$  e quindi non può aver generato il messaggio 3. Visto che  $R_B$  è scelto in modo casuale da uno spazio molto grande (un numero casuale di 128 bit) ed molto improbabile che Trudy possa aver preso  $R_B$  e la sua risposta da una sessione precedente. È molto improbabile che Trudy possa indovinare la risposta corretta alla richiesta contenuta nel messaggio 2.
4. A questo punto Bob è sicuro di essere in comunicazione con Alice, ma Alice non è ancora sicura di niente. Per quanto ne sa Alice, Trudy potrebbe aver intercettato il messaggio 1 e aver inviato la risposta  $R_B$ . Magari Bob è morto la notte prima. Per scoprire con chi sta parlando, Alice sceglie un numero casuale  $R_A$  e lo invia a Bob in chiaro nel messaggio 4.

5. Quando Bob risponde con  $K_{AB}(R_A)$ , Alice sa di essere veramente in comunicazione con Bob.
6. A questo punto è possibile stabilire una chiave di sessione: Alice ne sceglie una,  $K_S$ , e la manda a Bob cifrata con  $K_{AB}$ .

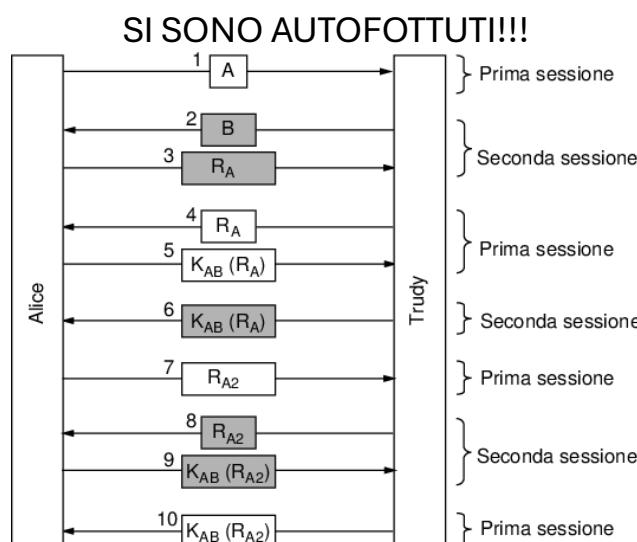
Il protocollo contiene cinque messaggi. È possibile ridurre il numero di messaggi scambiati facendo iniziare ad Alice la senza aspettare che sia Bob a farlo. Bob invia la sua richiesta mentre risponde a quella di Alice. L'intero protocollo si riduce quindi a tre messaggi, invece di cinque.



Essendo che non si conosce la chiave non sarebbe possibile cifrare i nounce. Se l'altra parte fosse un server sarebbe possibile un **attacco di riflessione** che consiste nell'aprire più comunicazioni.

1. Comincia con Trudy che finge di essere Alice e invia  $R_T$ , Bob risponde come al solito con una richiesta,  $R_B$ . Adesso Trudy è bloccata perché non conosce  $K_{AB}(R_B)$ .
2. Apre una seconda sessione con il messaggio 3, fornendo come richiesta la  $R_B$  presa dal messaggio 2. Bob cifra la  $R_B$  e invia  $K_{AB}(R_B)$  nel messaggio 4.

A questo punto Trudy ha l'informazione mancante e può completare la prima sessione e interrompere la seconda. Bob adesso è convinto che Trudy sia Alice, perciò quando questa domanda il suo estratto conto, Bob lo trasmette senza indugi. Quando Trudy chiede a Bob di trasferire tutti i soldi su un conto segreto in Svizzera, Bob eseguirà anche questo ordine senza esitazioni. La morale della storia è:



Le quattro regole generali seguenti sono spesso di aiuto.

1. Fare in modo che chi inizia l'autenticazione provi la sua identità prima che lo faccia chi risponde.
2. Fare sì che i due soggetti, cioè chi inizia l'autenticazione e chi risponde, usino due chiavi differenti per provare la loro identità, anche se ciò significa usare due chiavi condivise  $K$  e  $K'$ , da usare a seconda della direzione della comunicazione.
3. Far sì che i due soggetti utilizzino, per le richieste, numeri presi da insiemi diversi. Per esempio, chi inizia l'autenticazione può usare i numeri pari e chi risponde quelli dispari.
4. Rendere il protocollo resistente ad attacchi che coinvolgono una seconda sessione parallela, dove le informazioni ottenute da una sessione possono essere usate per l'altra.

Se anche una sola di queste regole non è rispettata, il protocollo può essere spesso forzato. Ma non è dei che, anche se tutte queste rispettar allora il protocollo sia completamente sicuro.

## PROTOCOLLO HMAC

Il **protocollo HMAC (Hashed Message Authentication Code)** è un meccanismo di autenticazione dei messaggi basato su una funzione di hash crittografica combinata con una chiave segreta. Per questo motivo, HMAC è talvolta indicato anche come **funzione hash con chiave**.

L'obiettivo principale dell'HMAC è **garantire l'integrità e l'autenticità di un messaggio** tra due parti che condividono una chiave segreta. In pratica:

- Il mittente prende in input la **chiave segreta** e il **messaggio da trasmettere**, e genera un **valore hash a lunghezza fissa**, chiamato **MAC** (Message Authentication Code), che viene inviato insieme al messaggio.
- Il destinatario, ricevuto il messaggio e il MAC, può **ricalcolare il valore hash** usando la stessa chiave segreta. Se il MAC calcolato corrisponde a quello ricevuto, si ha la **certezza che il messaggio non è stato alterato** e che proviene da chi dichiara di averlo inviato.

A differenza delle normali funzioni hash (come SHA-256), che non usano chiavi, HMAC è **resistente agli attacchi attivi**, perché **senza conoscere la chiave K è impossibile generare un MAC valido**, anche se si conosce il contenuto del messaggio.

Inoltre, l'utilizzo dell'HMAC **implica implicitamente l'identità del mittente**, poiché solo chi possiede la chiave segreta può produrre un MAC corretto.

La sicurezza di HMAC dipende principalmente dalla **robustezza della funzione di hash** sottostante (es. SHA-256 o SHA-3). I progettisti di HMAC hanno dimostrato formalmente che, anche se la funzione hash presenta alcune

debolezze (come nel caso di MD5 rispetto alle collisioni), HMAC **può comunque rimanere sicuro**, poiché:

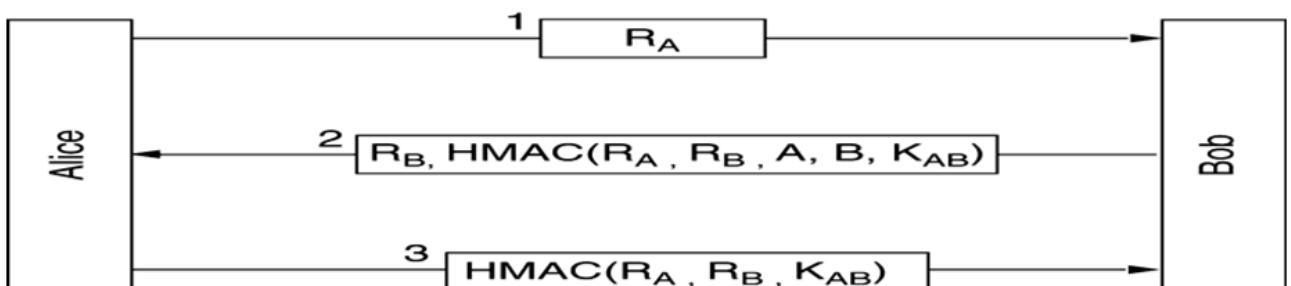
- Un attaccante **non può generare offline** coppie valide messaggio-MAC senza conoscere la chiave.
- L'attacco dovrebbe basarsi **sull'osservazione di molte sequenze di messaggi cifrati** con la stessa chiave, rendendo gli attacchi a forza bruta **computazionalmente molto costosi**.

Nel contesto di un protocollo di autenticazione, come Kerberos o uno scambio autenticato tra Alice e Bob, viene utilizzato un HMAC per garantire che i messaggi non siano stati alterati da terze parti. In questo scenario:

- Il sistema utilizza la **chiave segreta di sessione KAB**, condivisa tra Alice e Bob.
- Quando il messaggio viene inviato, il mittente calcola l'HMAC usando i nonce **RA** e **RB**, e gli identificatori **A** e **B**, insieme a **KAB**.
- Alla ricezione, Alice — che conosce sia i dati che la chiave — **ricalcola l'HMAC** e lo confronta con quello ricevuto. Se coincidono, l'autenticazione ha successo.

Dal punto di vista della sicurezza:

- **KAB non viene mai trasmessa**, quindi un attaccante come Trudy non può intercettarla né dedurla.
- Trudy **non può richiedere il calcolo di un digest su dati scelti**
- **Non può nemmeno calcolare un HMAC valido**, perché non ha accesso alla chiave segreta KAB.

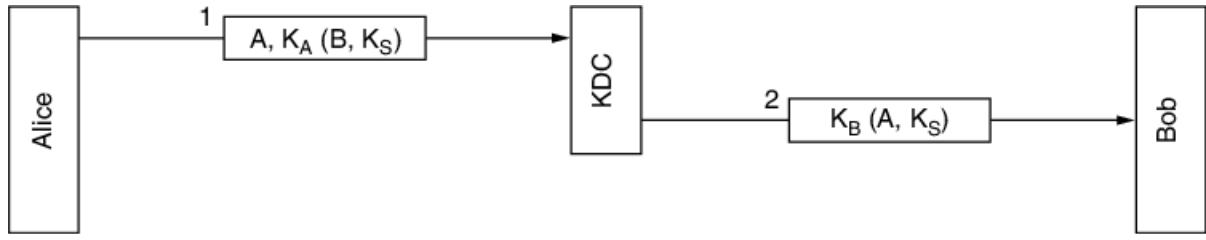


Ripensandoci meglio, dopotutto forse questo algoritmo non serve a nulla: per parlare con  $n$  persone in questo modo servono  $n$  chiavi.

Queste informazioni devono essere distribuite o scambiate in maniera sicura e devono essere mantenute segrete, ma questo può essere un problema.

Si può introdurre un centro di distribuzione delle chiavi o **KDC**. Ogni utente ha una singola chiave condivisa con il KDC e la gestione dell'autenticazione e della chiave di sessione sono gestite dal KDC.





1. Alice sceglie una chiave di sessione  $K_s$  e indica al KDC che vuole comunicare con Bob usando  $K_s$ .
2. Questo messaggio viene cifrato usando la chiave segreta che Alice condivide (solamente) con KDC,  $K_A$ .
3. Il KDC decifra il messaggio, estrae l'identità di Bob e la chiave di sessione, poi costruisce un nuovo messaggio contenente l'identità di Alice e la chiave di sessione, insieme spedisce il nuovo messaggio a Bob. Questa cifratura è fatta con  $K_B$ , la chiave segreta che Bob condivide con KDC: quando Bob decifrerà il messaggio, saprà che Alice desidera parlare con lui e saprà anche la chiave che Alice vuole usare.

L'autenticazione di Alice è garantita dal fatto che solo lei poteva inviare un messaggio cifrato con la sua chiave segreta condivisa con il KDC.

L'autenticazione del KDC è garantita a Bob, perché nessun altro poteva generare un messaggio cifrato con la chiave segreta che solo Bob e il KDC conoscono. Nessun attaccante esterno può conoscere o alterare la chiave di sessione, perché non ha accesso alle chiavi segrete individuali.

Trudy ha bisogno di soldi, quindi s'inventa qualche servizio legittimo che può fornire ad Alice, presenta una buona offerta e ottiene il lavoro. Dopo aver finito il lavoro, Trudy chiede educatamente ad Alice di essere pagata con un bonifico bancario. Alice stabilisce perciò una chiave di sessione con il suo banchiere, Bob, a cui invia il messaggio di richiesta di un bonifico a favore di Trudy.

Nel frattempo, Trudy è tornata alle sue vecchie abitudini e intercetta i messaggi sulla rete. Copia sia il messaggio 2, sia la successiva richiesta di bonifico. Più tardi Trudy manda nuovamente entrambi i messaggi a Bob. Bob li riceve e pensa: Alice deve aver assunto di nuovo Trudy, chiaramente lavora bene. Bob procede di nuovo al trasferimento della stessa somma di denaro dal conto di Alice a quello di Trudy. Dopo aver ricevuto la cinquantesima copia di messaggi uguali, Bob va a cercare Trudy per offrirle un bel prestito per espandere quello che è ormai un business di sicuro successo. Questo problema si chiama **attacco di ripetizione** (*replay attack*).

Per bloccare gli attacchi di ripetizione si può aggiungere al messaggio un timestamp in modo che quelli scaduti vengono immediatamente scartati dal

destinatario. Il problema di questo approccio è che su una rete gli orologi non sono mai sincronizzati; quindi, si deve prevedere un intervallo di tempo in cui il timestamp rimane valido.

Trudy può ripetere il messaggio durante questo intervallo e farla franca.



La seconda soluzione consiste nel mettere un nonce in ciascun messaggio. Ogni soggetto deve ricordarsi tutti i nonce precedenti e scartare ogni messaggio che contiene un nonce già usato. I nonce vanno ricordati per sempre, altrimenti Trudy può ripetere un messaggio vecchio di 5 anni.

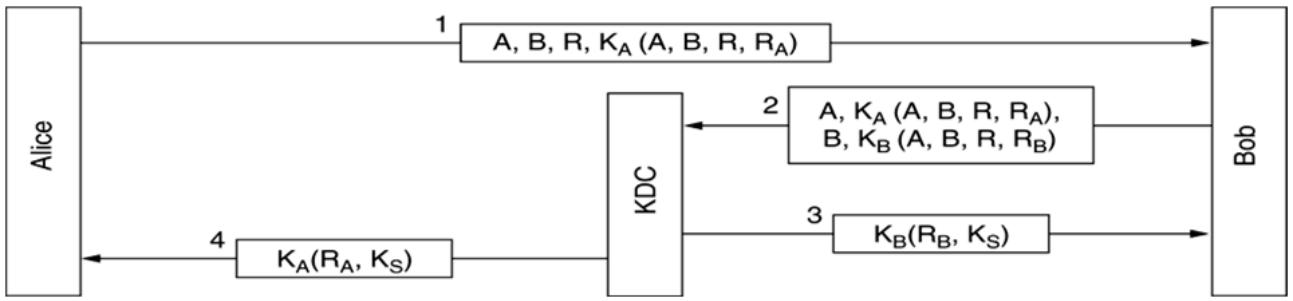
Inoltre, se qualche macchina riparte dopo un malfunzionamento e dimentica la lista dei nonce, può diventare vulnerabile ad attacchi di ripetizione. Timestamp e nonce si possono combinare per limitare l'intervallo di tempo in cui ricordare i nonce.

## Protocollo di Otway-Rees

Una soluzione può essere il **protocollo di Otway-Rees** che permette di stabilire una **chiave di sessione sicura** tramite una KDC utilizzando **solo crittografia simmetrica**.

1. Alice inizia generando una coppia di numeri casuali:  $R$  (che verrà usata come identificatore comune) e  $R_A$  (che Alice invierà come richiesta a Bob).
2. Quando Bob riceve questo messaggio, ne costruisce uno nuovo usando la parte cifrata del messaggio di Alice e una parte analoga costruita da lui stesso. Entrambe le parti, cifrate con  $K_A$  e  $K_B$ , identificano Alice e Bob, contengono l'identificatore comune e contengono un challenge.
3. Il KDC controlla che  $R$  abbia lo stesso valore in entrambe le parti. Potrebbe anche non essere vero, se Trudy ha manomesso  $R$  nel messaggio 1 oppure alterato parte del messaggio 2. Se le due  $R$  coincidono, il KDC considera valido il messaggio di richiesta di Bob, perciò genera una chiave di sessione e la cifra due volte: una per Alice e una per Bob.

Ogni messaggio contiene il numero casuale del ricevitore, come prova che il messaggio è stato generato dal KDC e non da Trudy. A questo punto Alice e Bob sono in possesso della stessa chiave di sessione e possono cominciare a comunicare. Al primo scambio di dati, ognuno dei due può vedere che l'altro è in possesso di un'identica copia di  $K_s$ , quindi l'autenticazione è completata.



## KERBEROS

Kerberos è una variante di Needham-Schroeder. La principale differenza rispetto a Needham-Schroeder sta nel fatto che Kerberos suppone che tutti gli orologi siano sincronizzati in modo abbastanza preciso.

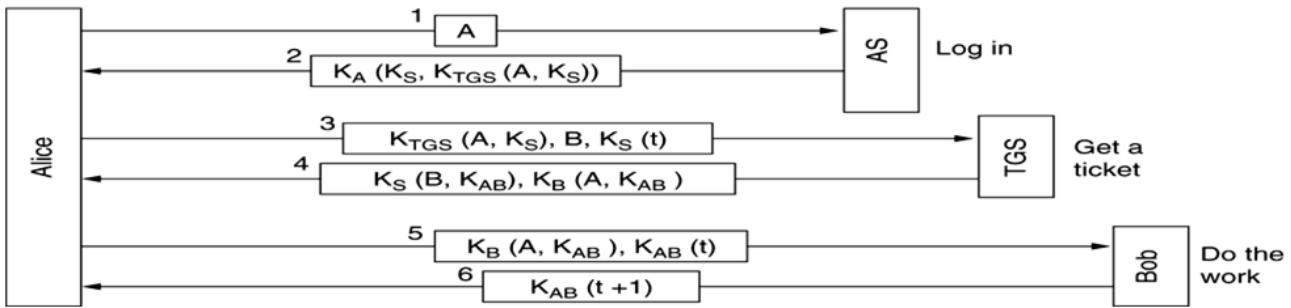
Kerberos utilizza tre server in aggiunta ad Alice:

- Authentication Server AS:** verifica l'identità degli utenti
- Ticket-Granting Server TGS:** emette ticket che provano l'identità di chi li possiede
- Il server Bob:** è quello che compie il lavoro che Alice ha richiesto.

AS è simile a un KDC, nel senso che condivide una password segreta con ogni utente. Il TGS ha il compito di emettere dei ticket da usare per convincere i server veri e propri del fatto che il loro possessore è realmente chi dichiara di essere.

1. Alice inizia una sessione da una qualunque workstation pubblica, inserendo il suo nome.
2. La workstation invia il nome di Alice in chiaro all'AS.
3. La risposta sarà formata da una chiave di sessione e da un ticket per il TGS,  $K_{TGS}(A, K_s, t)$ . La chiave di sessione è cifrata usando la chiave segreta di Alice K<sub>A</sub>, quindi solo Alice può decifrarla. Solo quando il messaggio 2 arriva la workstation chiede la password di Alice (e non prima).
4. La password viene usata per generare K<sub>A</sub> per decifrare il messaggio 2 e ottenere la chiave di sessione K<sub>s</sub>.
5. La workstation sovrascrive la password di Alice, assicurandosi così che sia rimasta all'interno della workstation stessa solamente per pochi millisecondi.

Se Trudy tenta di fare il login con il nome di Alice, la password che inserirà sarà sbagliata e la workstation potrà accorgersene in quanto la parte standard del messaggio 2 non sarà corretta.



Dopo il login, Alice può chiedere alla workstation di contattare il file server Bob.

6. La workstation invia il messaggio 3 al TGS chiedendogli un ticket da usare con Bob. L'elemento chiave di questa richiesta è  $K_{TGS}(A, K_s, t)$ , che è cifrato con la chiave segreta del TGS ed è usato come prova che il mittente è Alice.
7. Il TGS risponde al messaggio 4 creando una chiave di sessione  $K_{AB}$ , che Alice potrà usare con Bob. Ne vengono spedite indietro due versioni: la prima è cifrata solamente con  $K_s$ , così che Alice la possa leggere, mentre la seconda è cifrata con la chiave di Bob  $K_B$  così che Bob riesca a leggerla.

Trudy può copiare il messaggio 3 e provare a usarlo di nuovo, ma sarà bloccata dal timestamp cifrato  $t$ , che è inviato insieme al messaggio. Trudy non può rimpiazzare il timestamp con uno più recente, in quanto non conosce  $K_s$ , la chiave di sessione che Alice usa per parlare con il TGS. Se Trudy rimanda il messaggio 3 molto velocemente, l'unica cosa che otterrebbe sarebbe un'altra copia del messaggio 4, che non è riuscita a decifrare la prima volta e che quindi non sarà neppure in grado di decifrare questa volta.

Adesso Alice può inviare  $K_{AB}$  a Bob e stabilire una sessione con lui; anche questo scambio è accompagnato da un timestamp. Per Alice la risposta è una prova del fatto che sta parlando con Bob e non con Trudy.

8. Alice riesce a parlare con Bob usando  $K_{AB}$ , inviandola con  $K_B(A, K_{AB}, t)$
9. Bob autentica Alice grazie al token che solo lui può decifrare e che contiene la chiave di sessione  $K_{AB}$  e l'identità di A
10. Bob, quindi, restituendo il timestamp aggiornato, si autentica, a sua volta, con Alice

**Accessi multipli al Realm** Un **realm** è un insieme di utenti e servizi che condividono la stessa autorità di autenticazione.

Con accesso multiplo si intende che un utente può accedere a più servizi (magari anche su macchine diverse) all'interno dello stesso realm, senza dover reinserire le credenziali ogni volta.

Il processo comincia quando l'utente, per esempio Alice, accede al sistema e fornisce le sue credenziali. Se l'autenticazione ha successo, il KDC le rilascia un

**Ticket Granting Ticket** (TGT). Questo ticket ha una durata limitata, finché è valido, Alice può usarlo per ottenere l'accesso ad altri servizi all'interno del realm, senza dover reinserire la password.

Quando Alice desidera accedere a un servizio specifico, come un server e-mail o un file server, non deve autenticarsi di nuovo, basta presentare il TGT al Ticket Granting Service (TGS), che fa parte del KDC, e riceverà in risposta un ticket di servizio. Questo nuovo ticket contiene tutte le informazioni necessarie per stabilire una connessione sicura con il servizio richiesto, compresa una chiave di sessione condivisa.

Se vuole accedere a realm che non appartengono alla sua organizzazione si parla di **inter-realm authentication**. I KDC dei diversi realm devono avere una relazione di fiducia predefinita.

Se Alice è nel realm A e vuole accedere a un servizio nel realm B:

1. Il KDC di A le dà un ticket per il KDC di B.
2. Alice usa quel ticket per ottenere un ticket per il servizio nel realm B.

Questo meccanismo consente accesso federato tra domini diversi.

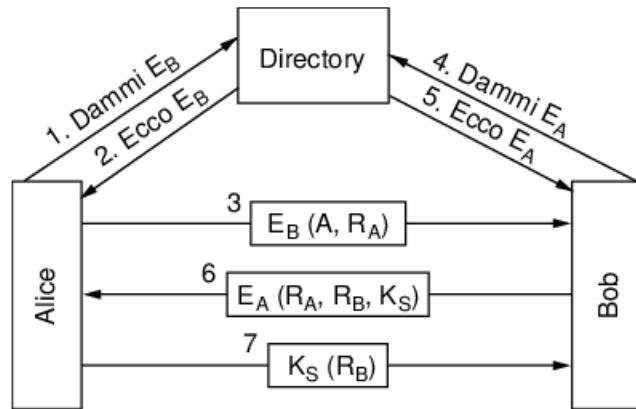
## Autenticazione con crittografia a chiave pubblica

La mutua autenticazione si può eseguire anche usando la crittografia a chiave pubblica.

1. Alice ha bisogno della chiave pubblica di Bob. Se esiste una PKI con un directory server che fornisce i certificati per le chiavi pubbliche, Alice gli può chiedere il certificato di Bob, per il messaggio 1.
2. La risposta, il messaggio 2, è un certificato X.509 che contiene la chiave pubblica di Bob.
3. Alice verifica la correttezza della firma, quindi manda a Bob un messaggio 3 che contiene la sua identità e un nonce.
4. Quando Bob riceve questo messaggio non sa ancora se proviene da Alice o da Trudy, ma prosegue chiedendo al directory server la chiave pubblica di Alice, messaggio 4, che riceve (messaggio 5).
5. Quindi Bob manda ad Alice il messaggio 6 che contiene l' $R_A$  di Alice, il suo nonce  $R_B$  e una proposta per la chiave di sessione,  $K_S$ .
6. Alice riceve il messaggio 6 e lo decifra usando la sua chiave privata. Vede che contiene  $R_A$ , e capisce che il messaggio viene da Bob, visto che Trudy non ha modo di determinare  $R_A$ . Inoltre, deve essere un messaggio fresco e non una ripetizione, in quanto Alice ha appena inviato  $R_A$  a Bob.
7. Alice conferma di essere d'accordo sulla chiave di sessione con il messaggio 7.

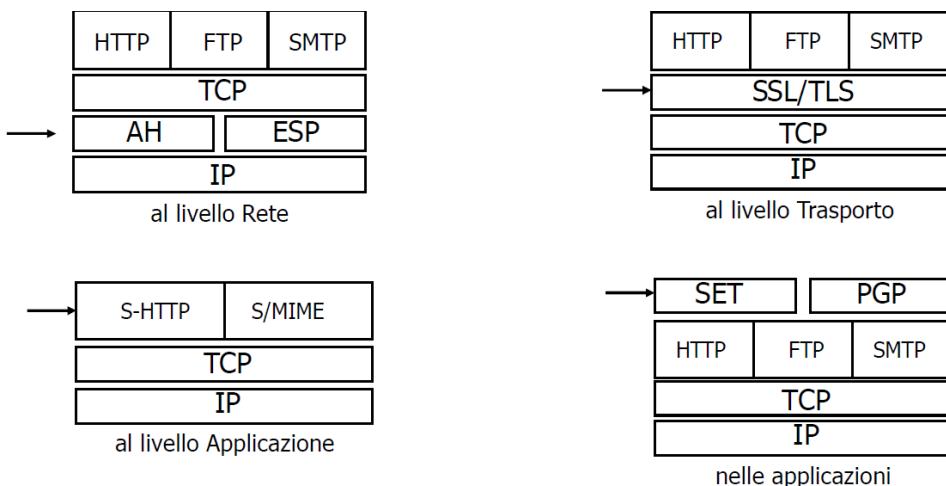
8. Quando Bob riceve il messaggio e vede che contiene l' $R_B$  cifrato con la chiave di sessione che ha appena generato, sa che Alice ha ricevuto correttamente il messaggio 6 e ha verificato  $R_A$ .

Che cosa può fare Trudy per cercare di forzare questo protocollo? Può fabbricare il messaggio 3 e cercare d'ingannare Bob facendogli spedire un messaggio ad Alice. Alice, però, vedrà un  $R_A$  che non ha spedito e quindi non procederà ulteriormente. Trudy non può falsificare il messaggio 7 di ritorno a Bob, perché non conosce né  $R_B$  né  $K_S$ , e non può saperli senza avere la chiave privata di Alice. Quindi Trudy non riesce a fare niente.



## Comunicazioni sicure e interazioni

Anche il modello ISO/OSI, per la comunicazione in rete, può essere reso sicuro nei suoi vari livelli, andando ad estendere il protocollo con altri layer che implementano la sicurezza.



### TLS/SSL

Fra il livello applicativo e di trasporto possono essere posti due protocolli per creare comunicazioni sicure con le socket; il **TLS Transport Layer Secure** che ha sostituito l'**SSL Secure Socket Layer**. Permettono

- la negoziazione di parametri per la connessione sicura tra client e server
- l'autenticazione mutua tra client e server
- la comunicazione segreta
- l'integrità dei dati scambiati

Dal lato sender, SSL/TLS riceve i dati da un'applicazione, li cifra e li indirizza ad una socket TCP; dal lato receiver, SSL/TLS legge i dati da una socket TCP, li decifra e li indirizza all'applicazione.

### Architettura di TLS

TLS costituito da due livelli di protocolli:

- **TLS Record Protocol** fornisce servizi di sicurezza di base a vari protocolli di livello superiore. In particolare, il Hypertext Transfer Protocol (HTTP), che fornisce il servizio di trasferimento per le interazioni Web client/server, può operare sopra TLS.
- tre protocolli di più alto livello: l'**Handshake Protocol**, il **Change Cipher Spec Protocol**, l'**Alert Protocol** e l'**Heartbeat Protocol**. Questi protocolli specifici di TLS vengono utilizzati nella gestione degli scambi.

**Connessione** – Una connessione è una forma di trasporto che fornisce un determinato tipo di servizio. Per TLS, tali connessioni sono collegamenti peer-to-peer. Le connessioni sono transitorie e ad ogni connessione è associata a una.

**Sessione** – un’associazione fra un client e un server. Le sessioni vengono create dall’Handshake Protocol. Le sessioni definiscono un insieme di parametri di sicurezza crittografica che possono essere condivisi fra più connessioni. Le sessioni vengono utilizzate per evitare l’onerosa negoziazione di nuovi parametri di sicurezza per ciascuna connessione.

Tra ogni coppia di parti, possono esservi più connessioni sicure. A ogni sessione vengono associati più stati. Una volta stabilita una sessione, vi è uno stato operativo corrente per la lettura e la scrittura (ovvero per la ricezione e l’invio). Inoltre, durante l’Handshake Protocol, vengono creati degli stati provvisori per la lettura e la scrittura. Alla conclusione dell’Handshake Protocol, gli stati provvisori diventano stati correnti.

Lo stato di una sessione è definito dai seguenti parametri:

- Identificativo di sessione – Una sequenza di byte arbitraria scelta dal server per identificare lo stato di una sessione attiva o riattivabile.
- Certificato del peer – Il certificato X509.v3 del peer. Questo elemento dello stato può essere nullo
- Metodo di compressione – L’algoritmo utilizzato per comprimere i dati prima della cifratura.
- Specifiche di cifratura – Specifica l’algoritmo di cifratura dei dati (come null, AES, ecc.) e l’algoritmo hash (come MD5 o SHA-1) usato per il calcolo del MAC. Inoltre, definisce gli attributi crittografici, come hash\_size.
- Segreto master – Un codice segreto di 48 byte condiviso tra il client e il server.
- Ripristinabile – Flag che indica se la sessione può essere usata per iniziare nuove connessioni

Il Record Protocol del TLS fornisce due servizi per le connessioni TLS:

**Confidenzialità** – L’Handshake Protocol definisce una chiave segreta condivisa che viene utilizzata per la cifratura convenzionale dei payload TLS.

**Integrità del messaggio** – Il protocollo Handshake definisce anche una chiave segreta condivisa che viene utilizzata per creare un codice MAC (Message Authentication Code)

## **Handshake protocol**

è l'Handshake Protocol. Questo protocollo permette al server e al client di autenticarsi reciprocamente e di negoziare un algoritmo di cifratura e di MAC, nonché le chiavi crittografiche da usare per proteggere i dati inviati in un record TLS. L'Handshake Protocol viene utilizzato prima della trasmissione di qualsiasi dato di livello applicativo

### **FASE 1-Inizializzazione delle funzioni di sicurezza**

Viene avviata la connessione logica fra client e server, scambiandosi informazioni sulla versione del protocollo TLS, id di sessione, algoritmi di cifratura e compressione da usare. Invia anche un nonce (Ra)  
Il server risponde con le sue scelte e manda un nonce (Rb)

### **FASE 2-Autenticazione del server**

Il server manda il proprio certificato in modo che il client possa ottenere dal certificato la chiave pubblica. Se il certificato non è firmato da un'autorità nota, manda la catena di certificati per risalire ad essa. Il browser usa un elenco di CA per verificare che la chiave del server sia autentica e viene memorizzata in un repo di chiavi pubbliche delle CA.

### **FASE 3-Authenticazione del client e sessione cifrata**

L'autenticazione del client permette ad un server di confermare l'identità di un utente. Usa lo stesso meccanismo previsto per autenticare il server.  
Successivamente, il client invia al server un'informazione chiave cifrata con algoritmo a chiave pubblica, mediante la quale costruire una chiave di sessione.

Il client risponde scegliendo una chiave casuale di 384 bit, detta **chiave di premaster** che invia al server in modo cifrato usando la chiave pubblica ottenuta dal certificato.

La **chiave di sessione** vera e propria è il risultato di un complesso calcolo basato sulla chiave di premaster, in combinazione con i due nonce.  
Entrambi la possono calcolare solo dopo che il client ha scelta la chiave di premaster. Una volta che entrambi l'hanno calcolata, con un complesso algoritmo, la fase di handshake si conclude e si passa

### **FASE 4- Cifratura della sessione**

Il Record Protocol accetta il messaggio di un'applicazione da trasmettere, frammenta i dati in blocchi di dimensioni di 16KB, optionalmente comprime i dati.

Si calcola la chiave segreta a partire dai due nonce e dalla chiave di premaster: questa chiave viene concatenata al testo in chiaro e il tutto è passato attraverso

l’algoritmo di hash negoziato in precedenza. Tale hash viene aggiunto a ogni frammento come MAC (message authentication code); quindi il frammento compresso e il suo MAC vengono cifrati con l’algoritmo di crittografia simmetrica negoziato in precedenza (XOR con il keystream RC4). Infine, viene aggiunta un’intestazione a ogni frammento e si procede alla sua trasmissione tramite la connessione TCP.

Gli attacchi si possono suddividere in quattro categorie principali:

### **1. Attacchi all’Handshake Protocol**

Già nel 1998 sono stati identificati attacchi che sfruttavano debolezze nell’uso di RSA durante la fase di handshake (la fase iniziale di scambio di chiavi tra client e server). Col tempo, questi attacchi sono stati raffinati per aggirare le difese introdotte e renderli più rapidi ed efficaci.

### **2. Attacchi ai dati e ai protocolli di record**

Alcuni attacchi hanno colpito direttamente i dati scambiati durante la connessione. Ad esempio:

- **BEAST (2011)**: un attacco pratico che sfruttava una debolezza teorica nella cifratura CBC usata da TLS. Permetteva di decifrare parte delle comunicazioni tramite attacchi a testo in chiaro scelto.
- **CRIME (2012)**: mirava a rubare dati sensibili (come cookie di autenticazione) sfruttando la compressione dei dati combinata con TLS. Poteva portare al dirottamento delle sessioni web.

### **3. Attacchi all’infrastruttura PKI (Public Key Infrastructure)**

Anche il sistema di gestione dei certificati digitali, essenziale per verificare l’identità dei server, è stato vulnerabile. Studi hanno mostrato che molte librerie usate per gestire SSL/TLS (come OpenSSL, GnuTLS, cURL, PHP, ecc.) contenevano errori che potevano compromettere la validazione dei certificati, aprendo la porta ad attacchi man-in-the-middle.

### **4. Altri tipi di attacchi**

Esistono anche attacchi meno classificabili, come quello DoS (Denial of Service) lanciato nel 2011 dal gruppo “The Hackers Choice”. Questo tipo di attacco sfrutta il carico computazionale della fase di handshake per mandare in crisi i server, sovraccaricandoli con connessioni ripetute o richieste di rinegoziazione TLS.

### **TLS 1.3: un passo avanti in sicurezza ed efficienza**

Nel 2014, il gruppo di lavoro IETF ha iniziato a sviluppare **TLS 1.3**, con l'obiettivo principale di aumentare la **sicurezza del protocollo** rispetto alle versioni precedenti. Al momento della stesura del documento originale, la nuova versione era ancora in fase di bozza, ma già conteneva modifiche sostanziali rispetto a TLS 1.2.

Uno dei cambiamenti più importanti è stata la **rimozione di molte funzionalità considerate obsolete o insicure**, per semplificare il protocollo e ridurre le superfici di attacco. Le funzioni eliminate includono:

- La **compressione dei dati**, che poteva esporre informazioni sensibili (come nel caso dell'attacco CRIME);
- **Cifrari non autenticati**, cioè quelli che non garantiscono integrità oltre alla riservatezza;
- L'uso di **RSA o Diffie-Hellman statico** per lo scambio delle chiavi;
- Il protocollo **Change Cipher Spec**, usato per cambiare i parametri di cifratura;
- **Algoritmi insicuri** come **RC4, MD5 e SHA-224**;
- Il **timestamp a 32 bit** nel messaggio iniziale di handshake (client\_hello);
- La possibilità di effettuare una **rinegoziazione** durante una connessione attiva, che poteva introdurre vulnerabilità.

Un altro importante cambiamento riguarda il metodo di **scambio delle chiavi**.

TLS 1.3 adotta esclusivamente algoritmi **Diffie-Hellman** (DH ed ECDH) per garantire la cosiddetta "**forward secrecy**": ogni sessione utilizza una chiave temporanea diversa, quindi anche se una chiave privata venisse compromessa in futuro, i dati delle sessioni precedenti rimarrebbero protetti.

Infine, TLS 1.3 introduce anche un **handshake più veloce**, che richiede **solo un round trip (1-RTT)** tra client e server per stabilire una connessione sicura.

Questo è possibile perché il client può inviare già i propri parametri crittografici nel primo messaggio, permettendo al server di calcolare subito le chiavi necessarie. Il risultato è una connessione più rapida e con minori opportunità per un attaccante di interferire.

---

## **IPSECURE**

---

L'IpSec è lo strato che va estendere aggiungendo la sicurezza al livello IP.

Se deve essere usato è necessario che tutti i dispositivi che intervengono nella comunicazione siano in grado di comprenderlo. L'uso di IPsec permette di garantire l'integrità, l'autenticità, cifratura e protezione da attacchi di replica.

## Architettura IPSec

IPSec consiste in una coppia di protocolli per implementare la sicurezza a livello di rete che sono

- l'**Authentication Header AH** che fornisce i servizi di autenticazione della sorgente e integrità dei dati. È stato deprecato e sostituito con
- l'**Encapsulation Security Payload ESP** che fornisce sia i servizi di AH e la segretezza dei dati tramite cifratura.

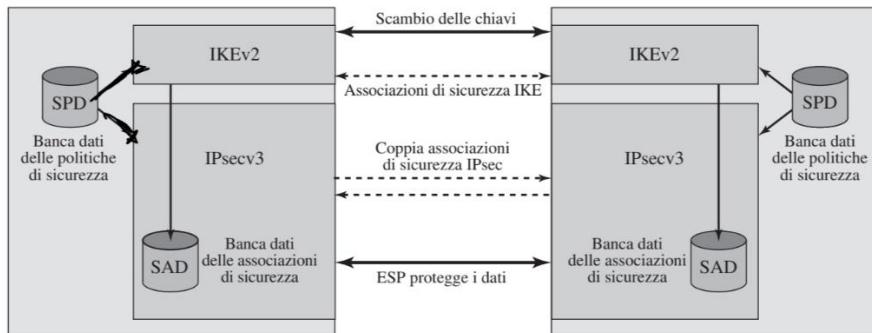
Inoltre, ha bisogno anche di un'astrazione per potersi collegare al protocollo e fornire i servizi di sicurezza, questo avviene tramite **Security Association SA**. Queste intervengono prima di inviare il datagram. Si ha una fase di handshake con cui viene creata la connessione logica, SA, e trasforma lo strato di rete senza connessioni in uno con connessioni logiche. La SA è unidirezionale, per la bidirezionalità devono essere stabilite due SA. Una SA è univocamente identificata da:

- l'indirizzo IP sorgente
- l'identificatore a 32 bit della connessione, detto **SPI (Security Parameter Index)**, assegnato all'associazione di sicurezza e con significato esclusivamente locale. Il valore SPI è trasportato negli header AH e ESP per permettere al sistema di destinazione di selezionare l'associazione di sicurezza in base alla quale elaborare il pacchetto ricevuto
- l'identificatore del protocollo (AH o ESP)

Serve un meccanismo di gestione delle chiavi detto **Internet Security Association and Key Management ISAKMP**. Definisce le procedure e il formato dei pacchetti necessari per stabilire, negoziare, modificare e terminare le Security Association (SA). Serve anche per selezionare il protocollo di sicurezza da utilizzare e i servizi da garantire (come la cifratura o l'autenticazione). Per lo scambio delle chiavi crittografiche, si affida a un protocollo dedicato, solitamente l'**Internet Key Exchange (IKE)**.

## Politiche di sicurezza

Le **politiche di sicurezza** sono applicate a ogni pacchetto IP che transita da una sorgente a una destinazione. La politica IPsec è determinata principalmente dall'interazione di due banche dati, la banca dati **Security Association Database SAD** e la banca dati **Security Policy Database SPD**.



## Flusso generale

1. I due nodi usano IKEv2 per negoziare e scambiare in modo sicuro le chiavi crittografiche. Queste chiavi serviranno per proteggere la comunicazione IPsec.
2. Vengono create e mantenute le associazioni di sicurezza per il protocollo IKE, che gestiscono lo stato dello scambio chiavi.
3. Dopo la negoziazione IKEv2, vengono stabilite le SA IPsec che effettivamente proteggono i dati. Sono solitamente due SA unidirezionali, una per ogni direzione.
4. IPsec utilizza il protocollo ESP (Encapsulating Security Payload) per cifrare e autenticare i dati durante la comunicazione.
5. Sia IKEv2 che IPsec consultano:
  - a. La **banca dati delle politiche di sicurezza (SPD)** per sapere quali dati devono essere protetti.
  - b. La **banca dati delle associazioni di sicurezza (SAD)** per recuperare i parametri di sicurezza da applicare.

## GESTIONE DELLE CHIAVI

I protocolli per la gestione automatizzata delle chiavi includono:

- **ISAKMP**: è il protocollo utilizzato per gestire le Security Associations (SA) e le chiavi. Fornisce un framework per la gestione delle chiavi su Internet e supporta protocolli specifici, inclusi i formati necessari per negoziare gli attributi di sicurezza delle SA.
- **Oakley**: è un protocollo dedicato alla determinazione delle chiavi. Si basa sull'algoritmo di Diffie-Hellman per lo scambio delle chiavi, ma offre un livello di sicurezza maggiore rispetto alla versione base.

**IKE** si occupa proprio di stabilire in modo sicuro le **Security Associations (SA)**, cioè gli accordi tra due nodi che definiscono come saranno protetti i dati scambiati.

Per fare questo, IKE:

- Scambia chiavi crittografiche usando il metodo Diffie-Hellman, permettendo alle due parti di creare una chiave segreta condivisa senza trasmetterla direttamente.
- Impiega un meccanismo per contrastare gli attacchi di intasamento, assicurando che il processo di negoziazione rimanga efficiente e non venga bloccato da attacchi DoS.
- Usa i **nonce** per prevenire attacchi di replay, garantendo che ogni messaggio sia unico e non riutilizzabile.
- Autentica lo scambio delle chiavi per difendersi dagli attacchi man-in-the-middle, assicurando l'identità delle parti coinvolte.
- Opera su segmenti UDP, favorendo una comunicazione veloce e leggera, che è importante per il rapido scambio di messaggi durante la fase di negoziazione.

## Security Associations Database

In ogni implementazione di IPsec esiste una banca dati chiamata **Security Associations Database (SAD)**, che contiene le informazioni e i parametri relativi a ciascuna associazione di sicurezza (SA). Ogni voce all'interno di questa banca dati descrive in modo dettagliato un'associazione di sicurezza specifica, attraverso diversi parametri fondamentali.

- **Security Parameter Index (SPI)**, un valore di 32 bit scelto dal destinatario dell'associazione di sicurezza. Questo valore serve a identificare in modo univoco l'associazione stessa. Nel caso di un'associazione di sicurezza in uscita, il SPI viene usato per costruire l'header AH (Authentication Header) o ESP (Encapsulating Security Payload) di ogni pacchetto. Per un'associazione in ingresso, invece, il SPI aiuta a mappare correttamente il traffico verso l'associazione di sicurezza corrispondente.
- **Sequence Number Counter**, anch'esso a 32 bit, che genera il campo “numero di sequenza” presente negli header AH o ESP. Questo campo è obbligatorio in tutte le implementazioni ed è usato per tracciare l'ordine dei pacchetti.
- Il **Sequence Counter Overflow** è un flag che segnala quando il contatore del numero di sequenza ha raggiunto il suo valore massimo (overflow). In questo caso, viene generato un evento che può impedire l'invio di ulteriori pacchetti con quell'associazione di sicurezza, contribuendo così a mantenere la sicurezza.
- La **finestra anti-replay (Anti-Replay Window)** è un altro meccanismo fondamentale che serve a verificare se un pacchetto in ingresso è stato già ricevuto, prevenendo così attacchi di replay.

- Per quanto riguarda la protezione dei dati, le informazioni relative all'**Authentication Header (AH)** comprendono l'algoritmo di autenticazione, le chiavi, la durata delle chiavi e altri parametri necessari per implementare AH, mentre quelle relative a **ESP** includono l'algoritmo di cifratura, l'algoritmo di autenticazione, le chiavi, i valori di inizializzazione e la durata delle chiavi.
- Il **tempo di vita dell'associazione di sicurezza (Lifetime)** definisce un intervallo temporale o un limite di dati (in byte) dopo il quale l'associazione deve essere sostituita da una nuova o chiusa. Viene indicata anche l'azione da eseguire a scadenza.
- la **modalità del protocollo IPsec**, che può essere tunnel, trasporto o una modalità generica
- la **Path MTU**, cioè la dimensione massima del pacchetto che può essere trasmesso senza frammentazione lungo il percorso, insieme alle informazioni sulla validità di questo valore.

## Security Policy Database

Il traffico IP viene associato alle specifiche SA tramite il SPD. Un SPD contiene delle voci, ognuna delle quali definisce un sottoinsieme del traffico IP e punta a un'associazione di sicurezza per tale traffico

Ogni voce della SPD è definita da un insieme di **selettori**, cioè criteri basati su campi IP e protocolli di livello superiore. Questi selettori permettono di filtrare il traffico in uscita e decidere quale associazione di sicurezza (SA) applicare a ciascun pacchetto IP.

### Selettori della SPD

Ogni voce della SPD può essere definita da uno o più dei seguenti selettori:

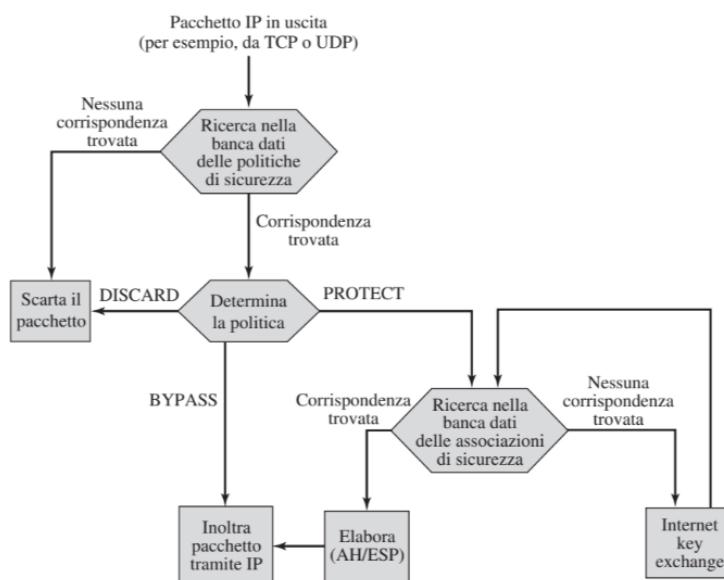
- **Indirizzo IP remoto**: singolo IP, intervallo, lista o maschera. Utile per gruppi di destinatari, ad esempio dietro un firewall.
- **Indirizzo IP locale**: stesso discorso dell'indirizzo remoto, ma per i mittenti.
- **Protocollo di livello superiore**: specifica il protocollo IP superiore (es. TCP, UDP, ICMP) tramite il campo *Protocol* (IPv4) o *Next Header* (IPv6). Può essere un valore preciso, "ANY" o "OPAQUE" (solo IPv6).
- **Porte locali e remote**: singole porte TCP/UDP, liste o caratteri jolly.
- **Nome utente**: identificatore dell'utente fornito dal sistema operativo, utile in ambienti integrati (non è presente negli header IP).

## Pacchetti in uscita

Un blocco dati del livello superiore, come TCP, viene passato al livello IP e viene creato un pacchetto IP, composto da un header IP e un corpo IP.

IPsec cerca nel SPD una corrispondenza con questo pacchetto.

- Se non viene trovata alcuna corrispondenza, il pacchetto viene scartato e viene generato un messaggio di errore.
- Se viene trovata una corrispondenza, la successiva elaborazione è determinata dalla prima voce corrispondente nel SPD.
  - Se la politica per questo pacchetto è **DISCARD**, allora il pacchetto viene scartato.
  - Se la politica è **BYPASS**, allora non vi è alcuna successiva elaborazione IPsec; il pacchetto viene inoltrato alla rete per essere trasmesso.
  - Se la politica è **PROTECT**, allora viene eseguita una ricerca nel SAD per trovare una voce corrispondente. Se non viene trovata nessuna voce, allora viene invocato IKE per creare un'associazione di sicurezza con le chiavi appropriate e vi si inserisce una voce. La voce corrispondente nel SAD determina l'elaborazione di questo pacchetto. Può essere eseguita la cifratura, l'autenticazione o entrambe, e può essere utilizzata la modalità **trasporto** o **tunnel**, dopodiché il pacchetto viene inoltrato alla rete per essere trasmesso.

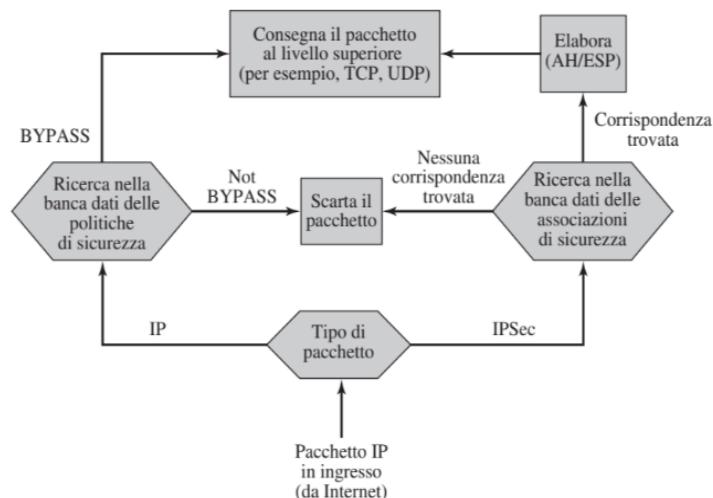


## Pacchetti in ingresso

Un pacchetto IP in ingresso innesca l'elaborazione IPsec.

IPsec determina, esaminando il campo **IP Protocol** (IPv4) o **Next Header** (IPv6), se si tratta di un pacchetto IP non protetto o di un pacchetto dotato di header/trailer ESP o AH.

- Se il pacchetto non è protetto, IPsec cerca nel SPD una corrispondenza per questo pacchetto.
  - Se la prima voce trovata ha politica **BYPASS**, l'header IP viene elaborato e rimosso, e il contenuto del pacchetto viene consegnato al livello immediatamente superiore, per esempio TCP.
  - Se la prima corrispondenza ha una politica **PROTECT** o **DISCARD**, o se non viene trovata nessuna voce corrispondente, il pacchetto viene scartato.
- Per un pacchetto protetto, IPsec controlla il SAD.
  - Se non viene trovata alcuna corrispondenza, il pacchetto viene scartato.
  - In caso contrario, IPsec applica l'elaborazione appropriata ESP o AH. Successivamente, l'header IP viene elaborato e rimosso, e il contenuto del pacchetto viene consegnato al livello immediatamente superiore, come il TCP.



IPsec può essere utilizzato in una modalità scelta fra le due possibili.

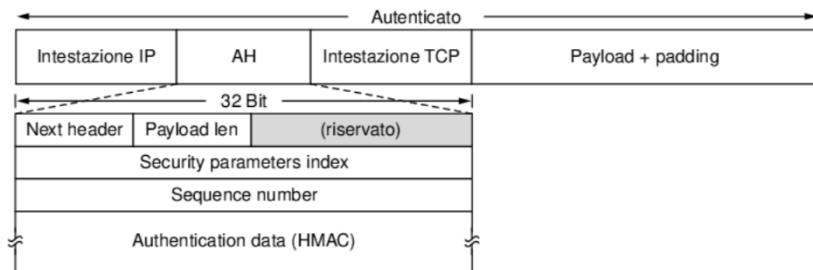
Nella **modalità trasporto** l'intestazione IPsec viene inserita subito dopo quella di IP. Il campo Protocol nell'intestazione IP è cambiato in modo da indicare che un'intestazione IPsec segue quella IP (prima dell'intestazione TCP).

L'intestazione IPsec contiene le informazioni di sicurezza: principalmente

l'identificatore di SA, un nuovo numero di sequenza, ed eventualmente un controllo d'integrità del payload.

Nella **modalità tunnel** l'intero pacchetto IP, intestazione compresa, viene encapsulato nel corpo di un nuovo pacchetto IP con un'intestazione IP completamente diversa. La modalità tunnel è utile quando il tunnel arriva in un punto diverso dalla sua destinazione finale.

## Formato pacchetto AH (modalità trasporto)



Il formato dei pacchetti dell'Authentication Header (AH) è un'intestazione aggiuntiva utilizzata per fornire autenticazione e controllo dell'integrità ai pacchetti IP, senza però garantire la segretezza (cifratura).

Ecco le sue caratteristiche principali:

- **Scopo** L'AH garantisce che un pacchetto ricevuto sia stato effettivamente trasmesso dal mittente indicato nell'intestazione e che non sia stato alterato durante il transito. Include anche l'indirizzo IP di origine nel controllo di integrità, rendendo impossibile per un intruso falsificare la provenienza del pacchetto. Viaggia in chiaro

### Campi dell'intestazione AH:

- **Next Header field**: Questo campo memorizza il valore originale del campo Protocollo IP che è stato sostituito con il valore 51 (che indica che segue un'intestazione AH). Ad esempio, per il TCP, il valore sarebbe 6.
- **Payload Length**: Contiene il numero di gruppi di 32 bit nell'intestazione AH meno 2.
- **Security Parameter Index (SPI)**: Questo è un identificatore di connessione inserito dal mittente. Viene utilizzato per indicare uno specifico record nel database del destinatario, il quale contiene la chiave condivisa e altre informazioni relative alla connessione.
- **Sequence Number**: Utilizzato per assegnare un numero a tutti i pacchetti inviati in una Security Association (SA). Utile per evitare attacchi di replay
- **Authentication Data**: Un campo a lunghezza variabile che contiene la firma digitale (o HMAC) del payload. Per esempio, un metodo comune consiste nel calcolare l'hash dei contenuti del pacchetto e della chiave insieme, utilizzando uno schema chiamato HMAC

Il controllo di integrità copre alcuni campi dell'intestazione IP, in particolare quelli che non cambiano durante la trasmissione da router a router. Ad esempio, l'indirizzo IP di origine è incluso, ma il campo Time to Live (TTL), che cambia a ogni salto di router, non lo è.

## Formato dei pacchetti ESP

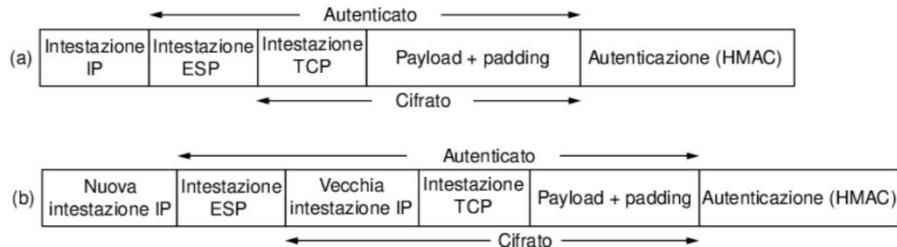


Figura 8.28 (a) ESP in modalità trasporto. (b) ESP in modalità tunnel.

Un pacchetto ESP è costituito da un'intestazione (header) e, optionalmente, un **trailer**, che incapsulano altri dati. Il formato include i campi:

- **Security Parameters Index (SPI)** (32 bit): Un valore assegnato all'associazione di sicurezza (SA) dal destinatario, utilizzato per identificare in modo univoco l'SA sulla base della quale elaborare il pacchetto ricevuto. Ha significato esclusivamente locale.
- **Sequence Number (Numero di Sequenza)** (32 bit): Un contatore che aumenta monotonicamente con ogni pacchetto inviato su una SA. È fondamentale per la funzionalità anti-replay. Il mittente non deve consentire che il numero di sequenza ricominci da zero dopo il valore  $2^{32} - 1$  per evitare pacchetti validi duplicati. Se il limite viene raggiunto, deve essere stabilita una nuova SA. Il destinatario implementa una "finestra" di dimensione fissa (di default 64) per tracciare i pacchetti validi ricevuti e scartare duplicati o pacchetti fuori finestra.
- **Initialization Value (IV) (opzionale)**: Se l'algoritmo di cifratura utilizzato richiede un vettore di inizializzazione, questo viene trasportato esplicitamente all'inizio del campo Payload Data e, generalmente, non viene cifrato.
- **Payload Data (Dati del Carico Utile)** (variabile): Contiene il segmento del protocollo di livello superiore (ad esempio TCP o UDP) in modalità trasporto, o l'intero pacchetto IP originale (intestazione IP inclusa) in modalità tunnel. Questo campo è protetto dalla cifratura.
- **Padding (Riempimento)** (0-255 byte): Questo campo ha diverse funzioni:
  - Assicurare che il testo in chiaro (Payload Data, Padding, Pad Length e Next Header) sia un multiplo della dimensione del blocco per gli algoritmi di cifratura a blocchi.

- Garantire che i campi Pad Length e Next Header siano allineati a destra all'interno di una parola di 32 bit, rendendo il testo cifrato un multiplo intero di 32 bit.
- (Non specificato nella fonte, ma implicito per alcuni usi) Aggiungere dati casuali per mascherare la reale lunghezza del payload.
- **Pad Length (Lunghezza del Riempimento)** (8 bit): Indica il numero di byte di riempimento che lo precedono.
- **Next Header (Intestazione Successiva)** (8 bit): Identifica il tipo di dati contenuti nel campo Payload Data, indicando la prima intestazione successiva (es. un'intestazione di estensione IPv6 o un protocollo di livello superiore come TCP).
- **Integrity Check Value (ICV) (Valore di Verifica dell'Integrità)** (variabile): Un campo di lunghezza variabile (multipla di 32 bit) che contiene il valore di controllo di integrità calcolato sul pacchetto ESP (senza contare il campo Authentication Data). Per gli algoritmi in modalità combinata (che gestiscono cifratura e integrità insieme), l'ICV può essere omesso e la verifica dell'integrità è gestita dall'algoritmo stesso. L'ICV è posizionato in coda al payload, un vantaggio per le implementazioni hardware in quanto può essere calcolato durante la trasmissione dei bit.

ESP supporta due modalità principali di utilizzo:

### 1. Modalità Trasporto:

- Fornisce protezione principalmente ai protocolli di livello superiore (es. TCP, UDP, ICMP). La protezione si estende al payload dei pacchetti IP.
- Utilizzata per la comunicazione end-to-end tra due host (es. client e server).
  - In IPv4, l'header ESP viene inserito immediatamente prima dell'header di livello trasporto, e il trailer ESP viene inserito dopo il payload. L'autenticazione Data viene aggiunta dopo il trailer ESP, se richiesta.
  - L'intero segmento del livello trasporto più il trailer ESP vengono cifrati. L'autenticazione copre il testo cifrato più l'header ESP. I router intermedi esaminano e processano l'header IP in chiaro, ma non hanno accesso al testo cifrato.

### 2. Modalità Tunnel:

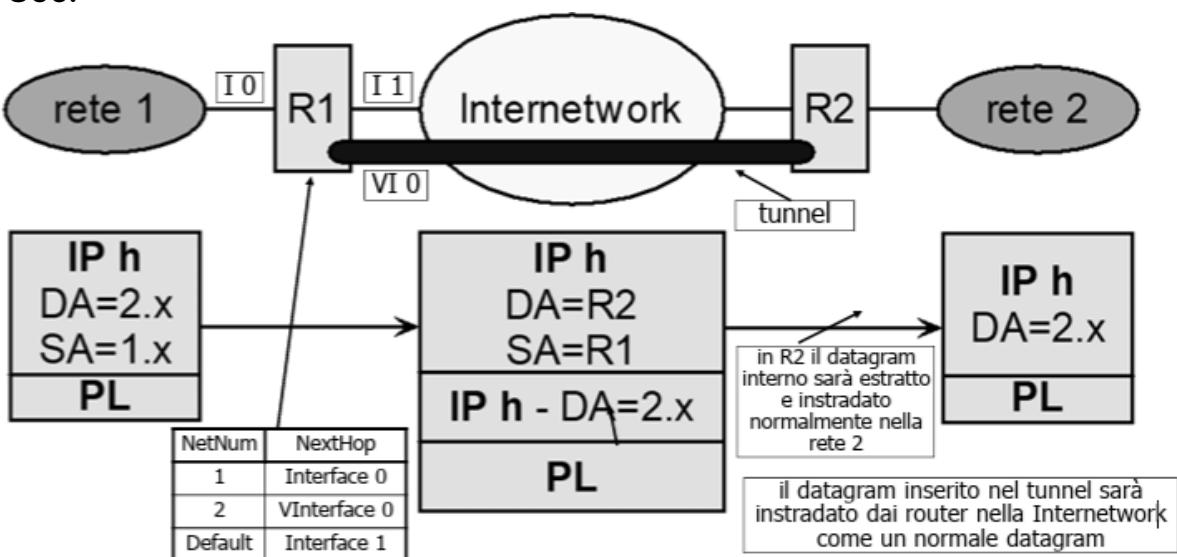
- Fornisce protezione all'intero pacchetto IP originale.
- L'intero pacchetto originale (con i campi AH o ESP aggiunti) viene trattato come il payload di un **nuovo pacchetto IP**, che avrà una nuova intestazione IP esterna.

- L'intero pacchetto originale, interno, attraversa un "tunnel" da un punto all'altro della rete IP; nessun router lungo il percorso è in grado di esaminare l'intestazione IP interna.
- Il nuovo pacchetto, più grande, può avere indirizzi sorgente e destinazione completamente diversi, aumentando il livello di sicurezza.
- In questa modalità, l'intero pacchetto IP interno (header interno + payload IP interno) più l'header/trailer ESP vengono cifrati e optionalmente autenticati. L'header IP esterno non è protetto.

**Confronto con AH** A differenza dell'Authentication Header (AH), che fornisce solo autenticazione e controllo dell'integrità, ESP offre anche la cifratura. Inoltre, mentre AH richiede di "bufferizzare" l'intero pacchetto per calcolare la firma prima della trasmissione, ESP posiziona l'HMAC (codice di autenticazione del messaggio) in coda al payload, permettendo un calcolo "on-the-fly" durante la trasmissione dei bit, il che è vantaggioso per le implementazioni hardware. Sebbene AH sia ancora incluso in IPsecv3 per la retrocompatibilità, il suo uso è stato deprecato perché ESP offre anche servizi di autenticazione.

## VPN

Una Virtual Private Network (VPN) è una rete privata, mappata su rete pubblica, costituita da **circuiti virtuali** che consentono di restringere la comunicazione ad un dato insieme di host. La comunicazione fra i dispositivi che fanno parte della VPN può avvenire solo attraversando certi nodi, che devono anche essi appartenere alla VPN. Si viene a creare una **connessione logica diretta** fra due host sopra al percorso fisico effettivo. Per creare le VPN si può sfruttare il livello di rete IP e si implementa realizzando la comunicazione con la modalità tunnel di IPSec.



Prima di attraversare il tunnel VPN che si tra fra due dispositivi che le appartengono, il pacchetto del mittente, viene incapsulato in un nuovo datagram IP il cui indirizzo di destinazione è il router all'estremità del tunnel, mentre l'IP sorgente è il router che fa da frontiera al mittente. Se le comunicazioni sono anche cifrate, la VPN diventa una sottorete sicura. Sfruttando il meccanismo del **tunneling** è difficile risalire a chi invia il messaggio e al destinatario, perché chi intercetta il messaggio vedrebbe gli IP dei router e il pacchetto originale è nascosto.

Nelle tabelle di routing del R1, è presente come nextHop una virtual interface che serve per poter stabilire a che deve essere inviato. Quando R2 riceverà il pacchetto, estrarrà il pacchetto originale e la comunicazione procede come al solito.

---

## HTTPS

---

Hyper Text Transfer Protocol Secure (HTTPS) è la versione sicura di HTTP. HTTPS cifra tutte le comunicazioni tra il browser e il sito web

I dati inviati utilizzando HTTPS forniscono tre importanti livelli di protezione:

Cifratura – Cifra i dati scambiati per proteggerli da eventuali intercettazioni.

Vengono cifrati l'URL del documento richiesto, il contenuto del documento, il contenuto dei moduli del browser (compilati dall'utente attraverso il browser), i cookie inviati dal browser al server e dal server al browser e il contenuto dell'intestazione HTTP.

Integrità dei dati – I dati non possono essere modificati o corrotti durante il trasferimento, intenzionalmente o meno, senza che le modifiche vengano rilevate.

Autenticazione – Garantisce che gli utenti comunichino con il sito web previsto. Protegge contro gli attacchi man-in-the-middle e aumenta la fiducia degli utenti, che si traduce in ulteriori vantaggi commerciali.

È possibile impiegare l'SHTTP secondo due modalità: • direttamente sullo strato del trasporto • su di un protocollo di trasporto sicuro

Questo non è un nuovo protocollo, si tratta di una Estensione di http. Consiste nell'andare ad incapsulare il messaggio http in **envelope**, contenitori che aggiungono informazioni sicure al messaggio http. Quindi il TCP riceve già info sicure. Offre le stesse garanzie di SSL/TLS, quindi averli entrambi è eccessivo. I requisiti di sicurezza possono essere negoziati tra client e server come richiesti, optional o rifiutati • I messaggi shttp sono costituiti da • la linea di richiesta/stato • è supportato il nuovo metodo “SECURE” • le linee di intestazione • ampliate rispetto a quelle previste per l'http • il corpo del

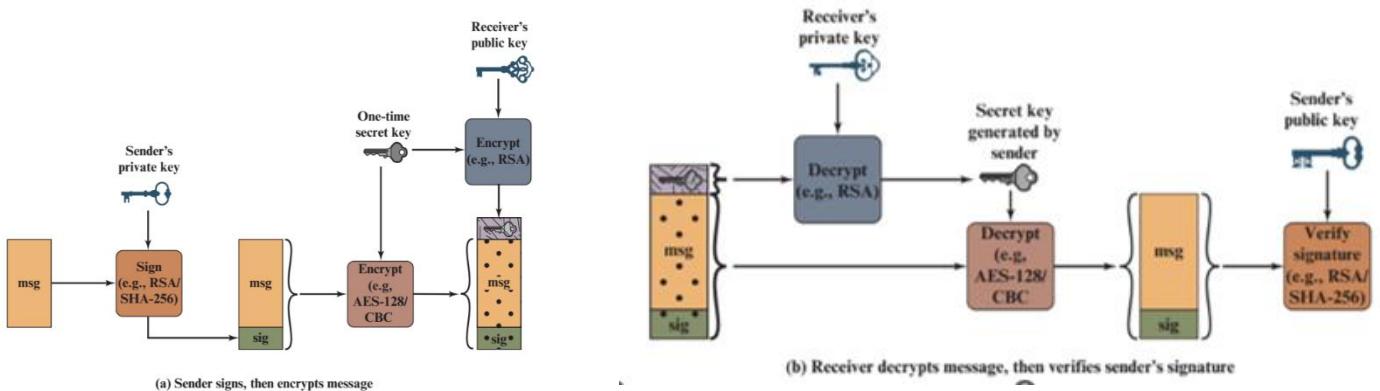
messaggio che può essere •un altro messaggio shttp (contenimento ricorsivo)  
 •un messaggio http •dati • I messaggi possono essere protetti con firma, autenticazione e cifratura, o con una combinazione di questi tre modi

## S/MIME

Secure/Multipurpose Internet Mail Extension (S/MIME) è un miglioramento della sicurezza dello standard di formato di posta elettronica Internet MIME basato sulla tecnologia di RSA Data Security Permette di firmare e/o cifrare i messaggi di posta elettronica.

**Finalità e Servizi Principali** S/MIME affronta diverse minacce alla sicurezza della posta elettronica, fornendo i seguenti servizi:

- **Autenticazione:** Garantisce l'identità del mittente. Questo viene realizzato tramite firme digitali, che utilizzano l'algoritmo RSA/SHA-256. Un codice hash del messaggio viene creato con SHA-256, cifrato con la chiave privata del mittente e incluso nel messaggio. Il destinatario decifra il digest del messaggio con la chiave pubblica del mittente e lo confronta con un digest calcolato indipendentemente.
- **Confidenzialità (Segretezza):** Protegge il contenuto del messaggio da divulgazioni non autorizzate. La confidenzialità è ottenuta cifrando i messaggi, solitamente con AES-128 in modalità CBC, utilizzando una



chiave di sessione monouso generata in modo pseudocasuale dal mittente per ogni messaggio. Questa chiave di sessione viene a sua volta cifrata con la chiave pubblica del destinatario e inclusa nel messaggio. Il destinatario recupera la chiave di sessione con la propria chiave privata e poi decifra il contenuto del messaggio. Non è necessario un protocollo di scambio chiavi di sessione tradizionale, dato che ogni messaggio è un evento indipendente.

- **Integrità dei Dati:** Assicura che il messaggio non sia stato alterato durante il trasferimento. Ciò è garantito dal codice hash (digital signature) che, se confrontato con il digest ricavato, rivela eventuali modifiche.
- **Non-Ripudio:** Fornisce la prova che il mittente ha inviato il messaggio, impedendogli di negarlo in seguito.
- **Compressione:** Offre la possibilità di comprimere un messaggio per risparmiare spazio durante la trasmissione e l'archiviazione. La compressione può essere applicata in qualsiasi ordine rispetto alle operazioni di firma e cifratura, sebbene sia sconsigliata per i dati binari già cifrati.
- **Compatibilità Email:** Permette di convertire il blocco cifrato (che può contenere octet arbitrari a 8 bit) in un flusso di caratteri ASCII stampabili (codifica a 7 bit), rendendolo compatibile con molti sistemi di posta elettronica esistenti che supportano solo testo ASCII. Questo processo è tipicamente realizzato tramite la conversione Base64. Anche se non cifrato, l'uso di Base64 può fornire un certo livello di riservatezza, rendendo il messaggio illeggibile per un osservatore occasionale.

### **Tipi di Contenuto dei Messaggi S/MIME**

S/MIME utilizza diversi nuovi tipi di contenuto MIME, designati con la denominazione PKCS (Public-Key Cryptography Standards). Questi includono:

- **Data:** Si riferisce al contenuto del messaggio codificato MIME interno, che può essere incapsulato in altri tipi.
- **SignedData:** Usato per applicare una firma digitale a un messaggio.
- **EnvelopedData:** Consiste in contenuto cifrato di qualsiasi tipo e chiavi di cifratura del contenuto cifrate per uno o più destinatari. La preparazione di un'entità envelopedData MIME prevede la generazione di una chiave di sessione, la sua cifratura con la chiave pubblica del destinatario, la creazione di un blocco RecipientInfo, e la cifratura del contenuto del messaggio con la chiave di sessione.
- **CompressedData:** Usato per applicare la compressione dei dati a un messaggio.
- **Clear Signing Data** S/MIME consente anche la "firma chiara" (clear signing), che utilizza il tipo di contenuto multipart con un sottotipo signed. Questo processo di firma non trasforma il messaggio da firmare, permettendo che il messaggio venga inviato "in chiaro". Ciò significa che i destinatari con capacità MIME, ma senza capacità S/MIME, possono comunque leggere il messaggio. Il messaggio multipart/signed è composto da due parti: la prima parte contiene il contenuto del messaggio (che, se non è a 7 bit, deve essere codificato con Base64 o quoted-printable), e la seconda parte è una firma distaccata (un oggetto

signedData con un campo di contenuto del messaggio vuoto), codificata Base64 e con tipo MIME application/pkcs7-signature

## DNSSec

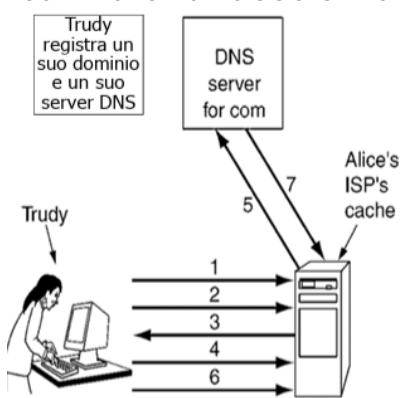
Quando digitiamo l'URL di una risorsa web, il sistema effettua una richiesta DNS per tradurre il nome simbolico nel corrispondente indirizzo IP del server che ospita quella risorsa. Questa operazione è fondamentale per indirizzare correttamente il traffico verso il server desiderato.

Tuttavia, il meccanismo DNS è soggetto ad attacchi, tra cui il DNS spoofing e DNS cache poisoning. In questo attacco, un malintenzionato falsifica l'indirizzo IP del mittente di una risposta DNS, in modo che sembri provenire da un server DNS legittimo.

Dato che il DNS utilizza UDP, un protocollo che non verifica l'origine delle risposte e non richiede l'instaurazione di una connessione, l'attaccante può inviare una risposta falsa a una query DNS, cercando di farla arrivare prima della risposta legittima. Se la risposta contraffatta viene accettata, il server DNS memorizza nella propria cache l'indirizzo IP errato. A quel punto, ogni utente che consulta quel resolver sarà reindirizzato verso un server controllato dall'attaccante, ad esempio un sito truffaldino simile all'originale, utile per sottrarre credenziali, diffondere malware o compiere frodi.

Questo attacco funziona perché le query e le risposte DNS sono correlate tramite un identificatore (ID), e molti resolver accettano la prima risposta valida che ricevono, senza verificare ulteriormente l'autenticità. In pratica, l'attaccante "indovina" l'ID e simula una risposta con IP malevolo prima che arrivi quella reale. Se l'attacco riesce, la cache del DNS viene avvelenata (poisoned), e il traffico viene deviato. Per effettuare l'attacco richiede che il truffatore si trovi nella stessa rete della vittima.

Il problema è che il truffatore deve azzeccare il numero di sequenza per realizzare l'attacco e individuarlo è abbastanza semplice.



**Trudy invia una richiesta DNS** al server DNS di una vittima per un nome appartenente a un dominio controllato da Trudy.

Il server DNS della vittima, non avendo l'informazione in cache, **contatta il server DNS autoritativo** del dominio di Trudy.

Quando il server di Trudy risponde, il server DNS della vittima **memorizza nella cache** l'associazione tra il nome e il server autoritativo di Trudy.

A questo punto, se la vittima effettua ulteriori richieste per nomi nello stesso dominio, il server DNS della vittima **contatterà direttamente il server di Trudy**. In questo modo, Trudy riesce a **intercettare la richiesta DNS legittima** e leggere l'ID (numero di sequenza) della query.

Subito dopo, Trudy **invia una nuova richiesta DNS** al server della vittima per un nome appartenente a un dominio legittimo (es. Bob).

Contemporaneamente, Trudy **invia molte risposte DNS false**, ognuna con un numero di identificazione leggermente diverso, maggiore o vicino a quello appena osservato.

Poiché il protocollo DNS utilizza **UDP**, non c'è un controllo rigoroso sull'origine della risposta: se una delle risposte false ha l'ID corretto e arriva prima di quella legittima, il server DNS della vittima la accetterà come valida.

Di conseguenza, il server DNS **memorizza nella cache l'informazione falsificata**, redirigendo gli utenti futuri verso server controllati da Trudy.

Questo attacco sfrutta:

- la **prevedibilità del numero ID** nelle richieste DNS;
- l'**assenza di autenticazione** nelle risposte DNS via UDP;
- la **latenza** tra la richiesta e la risposta legittima.

Il **DNSsec** è basato sulla crittografia a chiave pubblica ed è progettato per proteggere i client DNS dall'accettazione di record di risorse DNS alterati.

I servizi fondamentali offerti da DNSsec sono:

1. **Autenticazione dell'origine dei dati:** Garantisce che i dati provengano dalla fonte corretta e che siano stati approvati dal proprietario della zona.
2. **Verifica dell'integrità dei dati:** Assicura che il contenuto dei record di risorse non sia stato modificato durante il transito.
3. **Distribuzione della chiave pubblica:** Permette di memorizzare e utilizzare in modo sicuro le chiavi pubbliche.
4. **Autenticazione della transazione e della richiesta:** Protegge dagli attacchi di ripetizione o spoofing.

Ogni zona DNS (una porzione dello spazio dei nomi DNS) possiede una coppia di chiavi pubblica/privata. Tutte le informazioni inviate da un server DNS all'interno di una zona vengono firmate con la chiave privata della zona d'origine, permettendo al ricevente di verificarne l'autenticità.

La fiducia nella chiave pubblica (per la verifica della firma) della fonte è stabilita partendo da una zona affidabile (come la zona root) e costruendo una **catena di fiducia** fino alla fonte attuale della risposta. Questo si realizza attraverso verifiche successive della firma della chiave pubblica di un figlio da parte del suo genitore. La chiave pubblica della zona root, che è preconfigurata nei client DNS, è chiamata **trust anchor**. In questo modo, se un attaccante inserisse un

RRSet falso in una cache, un client sarebbe in grado di rilevare la contraffazione perché i contenuti della firma non sarebbero corretti.

I record DNS sono raggruppati in insiemi chiamati **RRSet** (resource record set), dove tutti i record di un insieme hanno lo stesso nome, classe e tipo. Per ogni RRSet, viene calcolato un hash crittografico e questo hash viene firmato con la chiave privata della zona (RSA). L'unità di trasmissione verso i client è l'RRSet firmato. Quando un client riceve un RRSet, può applicare la chiave pubblica della zona d'origine per decifrare l'hash ricevuto, calcolare il proprio hash e confrontare i due valori. Se i valori coincidono, i dati sono considerati validi e integri.

### Nuovi Tipi di Record DNSsec

- **DNSKEY**: Contiene una chiave pubblica di una zona, utente o host, specificando anche l'algoritmo crittografico utilizzato per la firma.
- **RRSIG**: È la firma digitale di un insieme di record di risorse (RRSet).
- **NSEC**: Fornisce una "negazione autenticata dell'esistenza", utilizzata per confermare in modo sicuro che un nome di dominio o un tipo di record non esiste all'interno di una zona, proteggendo da attacchi che cercano di indovinare nomi non esistenti.
- **DS (Delegation Signer)**: Facilita la firma delle chiavi e l'autenticazione tra le zone, contribuendo a creare la catena di autenticazione dal root dell'albero DNS fino a un nome di dominio specifico.

## Comunicazioni wireless

Alcuni dei fattori chiave che contribuiscono ad aumentare il rischio di sicurezza delle reti wireless rispetto alle reti cablate sono i seguenti:

- **Canale** – Le reti wireless in genere prevedono comunicazioni broadcast, le quali sono molto più esposte ad attacchi di intercettazione e alle interferenze di quanto non lo siano le reti cablate
- **Mobilità** – I dispositivi wireless sono più portabili e mobili dei dispositivi cablati.
- **Risorse** – Alcuni dispositivi wireless, come smartphone e tablet, dispongono di sistemi operativi sofisticati, ma di una memoria e risorse di elaborazione limitate con cui contrastare le minacce.
- **Accessibilità** – Alcuni dispositivi wireless, come sensori e robot, potrebbero rimanere incustoditi in luoghi isolati e/o a rischio. Ciò aumenta notevolmente la loro vulnerabilità agli attacchi fisici

### Componenti della rete IEEE 802.11 e modello architettonico

Il più piccolo elemento costitutivo di una wireless LAN è il **Basic Service Set BSS**, che consiste di stazioni wireless che eseguono lo stesso protocollo MAC e competono per l'accesso allo stesso mezzo trasmissivo wireless condiviso. Un BSS può essere isolato oppure può connettersi a un **Distribution System DS** dorsale tramite un **Access Point AP**.

L'AP funge da ponte e da ripetitore. In un BSS, le stazioni client non comunicano direttamente tra loro. Infatti, se una stazione nel BSS vuole comunicare con un'altra stazione nello stesso BSS, il frame MAC viene prima inviato dalla stazione sorgente all'AP e poi dall'AP alla stazione di destinazione. In maniera analoga, un frame MAC da una stazione nel BSS a una stazione remota viene inviato dalla stazione locale all'AP e poi ritrasmesso dall'AP lungo il DS per raggiungere la stazione di destinazione.

Il BSS corrisponde generalmente a ciò che in letteratura viene chiamato **cella**. Il DS può essere uno switch, una rete cablata oppure una rete wireless.

Quando tutte le stazioni nel BSS sono stazioni mobili che comunicano direttamente tra loro, il BSS è detto **Independent BSS IBSS**. Un IBSS in genere non è altro che una rete ad hoc. In esso tutte le stazioni comunicano direttamente e non vi è la necessità di coinvolgere un AP.

Nella Figura 13.5 viene mostrata una configurazione semplice in cui ciascuna stazione appartiene a un unico BSS, ossia ogni stazione si trova nel raggio di copertura wireless delle sole altre stazioni all'interno dello stesso BSS. Può anche succedere che due BSS siano sovrapposti a livello geografico, per cui una

singola stazione può far parte di più di un BSS. Inoltre, l'associazione tra una stazione e un BSS è dinamica. Le stazioni possono disattivarsi, entrare e uscire da un BSS.

Un **Extended Service Set ESS** consiste di due o più Basic Service Set interconnessi da un Distribution System. Al livello Logical Link Control (LLC), l'Extended Service Set viene visto come una singola LAN logica

## Proteggere le trasmissioni wireless

Le principali minacce alle trasmissioni wireless sono l'intercettazione, l'alterazione o l'aggiunta di messaggi e l'interruzione. Per contrastare l'intercettazione esistono due tipi di contromisure:

**Tecniche per nascondere il segnale** – Le organizzazioni possono adottare diverse misure per rendere più difficile per un attaccante individuare i loro access point wireless, tra cui la disattivazione della trasmissione in broadcast del service set identifier (SSID) da parte degli access point wireless, l'assegnazione di nomi criptici agli SSID, la riduzione della potenza del segnale al livello più basso possibile in grado di fornire ancora la copertura necessaria e il posizionamento degli access point wireless all'interno dell'edificio, lontano da finestre e pareti esterne. È possibile raggiungere una maggiore sicurezza tramite l'utilizzo di antenne direzionali e di tecniche di schermatura del segnale.

**Cifratura** – La cifratura di tutte le trasmissioni wireless è efficace contro l'intercettazione nella misura in cui le chiavi di cifratura vengono protette

## Proteggere gli access point wireless

La principale minaccia nei confronti degli access point wireless è l'accesso non autorizzato alla rete. L'approccio principale per impedire tale accesso è lo standard **IEEE 802.1X** per il controllo dell'accesso alla rete basato sulle porte. Lo standard fornisce un meccanismo di autenticazione per i dispositivi che desiderano collegarsi a una LAN o a una rete wireless.

## Architettura dei protocolli IEEE 802

Gli standard IEEE 802.11 vengono definiti entro la struttura di una serie di protocolli a più livelli.

**Livello fisico** – Il livello più basso che comprende funzioni quali la codifica/decodifica dei segnali e la trasmissione/ricezione di bit. Inoltre, il livello fisico include una specifica relativa al mezzo trasmittivo. Nel caso dell'IEEE 802.11, il livello fisico definisce anche le bande di frequenza e le proprietà delle antenne.

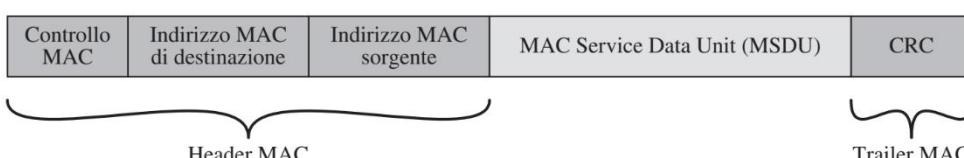
**Media Access Control** – Tutte le LAN sono composte da insiemi di dispositivi che condividono la capacità di trasmissione in rete. Per garantire un uso corretto ed efficiente di questa capacità occorre un qualche meccanismo di controllo dell'accesso al mezzo trasmissivo. Tale funzione è svolta dal livello Media Access Control (MAC) (letteralmente “controllo di accesso al mezzo”). Il livello MAC riceve i dati da un protocollo di livello superiore, in genere il livello Logical Link Control (LLC), sotto forma di un blocco di dati noto come MAC service data unit (MSDU). Di norma, il livello MAC svolge le seguenti funzioni:

- Durante la trasmissione assembla i dati in un frame, noto come MAC protocol data unit (MPDU), dotato dei campi indirizzo e rilevamento degli errori.
- Al momento della ricezione disassembra il frame ed svolge le operazioni di riconoscimento dell'indirizzo e rilevazione degli errori.
- Regola l'accesso al mezzo trasmissivo LAN.

Il formato del MPDU si differenzia in base ai vari protocolli MAC in uso. In genere, tutti gli MPDU presentano un formato simile che comprende i seguenti campi:

- **Controllo MAC** – Questo campo contiene tutte le informazioni di controllo del protocollo necessarie per il funzionamento del protocollo MAC. Ad esempio, si potrebbe specificare qui un livello di priorità.
- **Indirizzo MAC di destinazione** – L'indirizzo fisico di destinazione sulla LAN per questo MPDU.
- **Indirizzo MAC sorgente** – L'indirizzo fisico di origine sulla LAN per questo MPDU.
- **MAC Service Data Unit** – I dati dal livello superiore successivo. CRC – È il campo **Cyclic Redundancy**, conosciuto anche come campo **Frame Check Sequence FCS**. Si tratta di un codice di rilevamento degli errori simile a quello utilizzato in altri protocolli di controllo del livello collegamento. Il CRC viene calcolato sulla base dei bit dell'intero MPDU. Il mittente calcola il CRC e lo aggiunge al frame. Il destinatario esegue lo stesso calcolo sul MPDU ricevuto e lo confronta con il campo CRC di quel MPDU. Se i due valori non corrispondono significa che uno o più bit sono stati modificati durante la trasmissione.

I campi che precedono il campo MSDU costituiscono l'header MAC, mentre il campo che segue il campo MSDU è detto trailer MAC. L'header e il trailer contengono informazioni di controllo che accompagnano il campo dati e che vengono utilizzate dal protocollo MAC.



**Logical Link Control LLC** – Sono responsabili non solo del rilevamento degli errori tramite il CRC, ma anche del ripristino da tali errori ritrasmettendo i frame danneggiati. Nell’architettura del protocollo LAN queste due funzioni vengono suddivise tra i livelli MAC e LLC. Il livello MAC è responsabile del rilevamento degli errori e dell’eliminazione di eventuali frame contenenti errori. Il livello LLC tiene facoltativamente traccia di quali frame sono stati ricevuti in modo corretto e ritrasmette quelli che falliscono.

## Servizi IEEE 802.11

| Servizio         | Fornitore           | Utilizzato a supporto di      |
|------------------|---------------------|-------------------------------|
| Associazione     | Distribution System | Consegna di MSDU              |
| Autenticazione   | Stazione            | Accesso e sicurezza della LAN |
| Deautenticazione | Stazione            | Accesso e sicurezza della LAN |
| Disassociazione  | Distribution System | Consegna di MSDU              |
| Distribuzione    | Distribution System | Consegna di MSDU              |
| Integrazione     | Distribution System | Consegna di MSDU              |
| Consegna di MSDU | Stazione            | Consegna di MSDU              |
| Privacy          | Stazione            | Accesso e sicurezza della LAN |
| Riassociazione   | Distribution System | Consegna di MSDU              |

## FASI OPERATIVE

**Scoperta** – Un AP utilizza i messaggi detti Beacon e Probe Response per pubblicizzare la propria politica di sicurezza IEEE 802.11i. La STA li utilizza per identificare un AP per una WLAN con cui desidera comunicare. La STA si associa all’AP, di cui si serve per scegliere la suite di cifratura e il meccanismo di autenticazione quando i Beacon e Probe Response offrono una scelta.

**Autenticazione** – Durante questa fase la STA e l’AS dimostrano mutuamente la propria identità. L’AP blocca il traffico non finalizzato all’autenticazione tra STA e AS fino a quando la transazione di autenticazione non ha esito positivo. L’AP non partecipa alla transazione di autenticazione salvo che per l’inoltro del traffico tra la STA e l’AS.

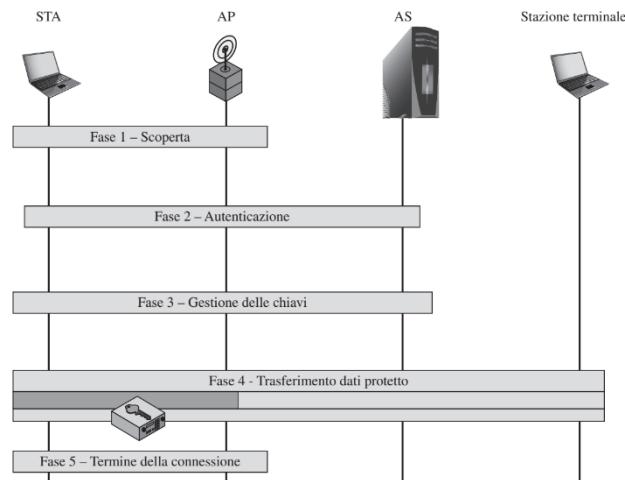
**Generazione e distribuzione delle chiavi** – L’AP e la STA eseguono diverse operazioni che permettono di generare e distribuire le chiavi crittografiche all’AP e alla STA. I frame vengono scambiati solamente tra l’AP e la STA.

**Trasferimento dati protetto** – I frame vengono scambiati tra la STA e la stazione terminale attraverso l'AP. L'ombreggiatura e l'icona relativa al modulo di cifratura stanno a indicare che il trasferimento sicuro dei dati riguarda solamente la STA e l'AP; non è prevista una protezione end-to-end.

### Termine della connessione – L'AP

e la STA si scambiano i frame.

Durante questa fase si interrompe la connessione protetta per poi ripristinare la connessione allo stato originale



Il **Wired Equivalent Privacy WEP** era lo schema crittografico originale incluso nello standard IEEE 802.11.

Il WEP mirava a cifrare le trasmissioni wireless per renderle confidenziali, dato che i segnali wireless, essendo broadcast, possono essere facilmente intercettati dai computer vicini non destinatari. Nel WEP, si presupponeva che tutti i dispositivi di rete condividessero una **chiave segreta precondivisa**. L'obiettivo dell'autenticazione nel WEP era di provare il possesso di questa chiave segreta da parte della stazione (STA).

Il processo di cifratura e integrità dei dati nel WEP si otteneva tramite l'algoritmo di **cifratura RC4**. Il trasmettitore di un'unità di dati MAC (MPDU) eseguiva i seguenti passi, noti come encapsulamento:

- Selezione di un IV (Initial Vector)**: Il trasmettitore sceglieva un valore per il vettore di inizializzazione.
- Generazione del seme RC4**: Il valore IV veniva concatenato con la chiave WEP condivisa tra trasmettitore e ricevitore per formare il seme, o chiave di input, per l'RC4.
- Cifratura**: Il testo in chiaro (il messaggio) veniva sottoposto a un'operazione XOR con l'output di un cifrario a flusso basato su RC4, generato dal seme.

**4. Controllo d'integrità:** Il WEP cifrava i dati per ottenere riservatezza ma utilizzava un meccanismo crittograficamente debole per il controllo d'integrità.

Il WEP era **imperfetto** e fu compromesso molto facilmente a causa

- **Meccanismo crittografico debole:** L'algoritmo RC4, sebbene non intrinsecamente insicuro, era utilizzato in modo errato. In particolare, il modo in cui il WEP utilizzava il vettore di inizializzazione (IV) rendeva possibile recuperare la chiave segreta. La riutilizzabilità degli IV con la stessa chiave segreta era una debolezza fatale.
- Anche se le chiavi degli utenti sono distinte, poiché sono impiegate per lungo tempo, sarebbe necessario utilizzare un IV diverso per ogni pacchetto inviato, in modo da evitare l'attacco classico allo stream RC4. Molte NIC, all'avvio, prevedono di inizializzare a zero l'IV e di incrementarlo ad ogni invio.

Se Trudy intercetta il traffico di un medesimo utente, riesce ad ottenere molti pacchetti cifrati con la stessa chiave e gli stessi IV.

Alcune NIC, invece, impiegano IV generati in maniera random.

Purtroppo, un IV è lungo 24 bit e ciò implica il riutilizzo di un medesimo IV ogni 5000 pacchetti inviati circa.

Se due pacchetti vengono cifrati utilizzando **la stessa chiave e lo stesso IV (Initialization Vector)**, è possibile risalire ai dati in chiaro semplicemente effettuando un'**operazione di XOR tra i due testi cifrati**. Questo tipo di vulnerabilità rende il sistema molto debole. Con un'analisi più approfondita, un attaccante può riuscire a **ricavare numerose coppie chiave-IV**, che possono poi essere **riutilizzate per generare pacchetti cifrati considerati validi all'interno della LAN**. In altre parole, l'attaccante potrebbe costruire traffico apparentemente legittimo, ma in realtà controllato. A rendere la situazione ancora più critica, va aggiunto che l'algoritmo **RC4**, utilizzato da WEP per la cifratura, presenta **diverse debolezze note**. Esistono attacchi in grado di **ricavare direttamente la chiave usata per cifrare il traffico**, rendendo completamente inutile la protezione offerta dal protocollo.

A causa di tutte queste problematiche, **alla fine del 2001 l'IEEE ha dichiarato il protocollo WEP insicuro**. Per sostituirlo sono stati introdotti protocolli più sicuri: **WPA (Wi-Fi Protected Access)** e successivamente **WPA2**, che migliorano significativamente la protezione delle reti wireless.

## **WPA**

Il **WPA** è stato introdotto come soluzione provvisoria per migliorare la sicurezza del WEP, senza richiedere la sostituzione dell'hardware esistente. Ha introdotto diversi miglioramenti fondamentali:

- **TKIP (Temporal Key Integrity Protocol)**: cambia la chiave di cifratura a ogni pacchetto, rendendo molto più difficile per un attaccante intercettare e decifrare il traffico.
- **Cifraatura**: uso di RC4 con chiave a 128 bit e IV a 48 bit
- **Controllo di integrità**: verificata con **Message Integrity Code MIC** o per i bro **Michael**, incluso un **frame counter** per evitare i replay attack

Il **WPA2** rappresenta l'evoluzione definitiva del WPA ed è diventato il nuovo standard di sicurezza per le reti Wi-Fi. A partire dal 2006, è stato reso **obbligatorio per ottenere la certificazione Wi-Fi**. WPA2 ha introdotto innovazioni significative per garantire una protezione molto più forte:

- **AES (Advanced Encryption Standard)**: WPA2 abbandona completamente RC4 e utilizza l'algoritmo AES, considerato uno dei più sicuri al mondo
- **CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)**: questo protocollo sostituisce TKIP e fornisce sia cifratura che integrità dei dati, rendendo le trasmissioni molto più difficili da manipolare o intercettare.

## **Sicurezza in Bluetooth**

Bluetooth ha un raggio di azione più corto di 802.11, quindi per essere attacchi tramite bluetooth bisogna essere molto vicini alla vittima.

Bluetooth versione 2.1 e seguenti hanno quattro modalità di sicurezza, che vanno dal nulla alla cifratura completa dei dati e il controllo dell'integrità che in generale è disabilitata.

Bluetooth presenta soluzioni di sicurezza su più livelli. A livello fisico, il **frequency hopping** fornisce un margine di sicurezza minimo. Visto che ogni dispositivo Bluetooth che si muove all'interno di una piconet deve essere informato della sequenza dei salti di frequenza, questa sequenza ovviamente non è segreta. La vera sicurezza inizia quando un nuovo dispositivo **slave** chiede un canale al **master**. Si suppone che i due dispositivi abbiano già effettuato uno scambio di chiave segreta condivisa contenuta già nei dispositivi. Queste chiavi condivise sono dette **passkey** che sono fisse in generale. Con l'accoppiamento semplice e sicuro di Bluetooth 2.1, i dispositivi scelgono un codice da un intervallo di sei cifre, negoziato tramite **Secure Simple Pairing**, che rende la passkey molto meno prevedibile ma ancora lontano dalla sicurezza.

Per stabilire un canale, master e slave effettuano il controllo per vedere se l'altro conosce la passkey. In caso affermativo, negoziano se il canale deve essere cifrato e se si deve effettuare il controllo di integrità o entrambe le cose. Quindi scelgono una chiave di sessione casuale a 128 bit, di cui alcuni bit possono essere resi pubblici. Lo scopo di questo indebolimento della chiave è di rendere il protocollo utilizzabile anche in quegli stati per cui la legge impone delle restrizioni all'esportazione o proibisce l'uso di chiavi più lunghe di quelle che le autorità governative riescono a forzare.

La cifratura usa uno **stream cipher** detto **E<sub>0</sub>** il controllo di integrità usa **SAFER+**. Entrambi sono cifrari a blocco tradizionali con chiave simmetrica. SAFER+ Il meccanismo di cifratura con lo stream cipher, il testo in chiaro viene applicato in XOR al keystream per generare il testo cifrato

Un altro punto riguardante la sicurezza è dato dal fatto che Bluetooth autentica solo i dispositivi e non gli utenti; quindi, il furto di un dispositivo Bluetooth può dare al ladro l'accesso ai dati finanziari, o comunque riservati, dell'utente. Bluetooth però implementa la sicurezza anche nei livelli superiori, quindi nel caso in cui ci sia una violazione a livello data link, rimane ancora un po' di sicurezza, specialmente per le applicazioni che richiedono l'inserimento manuale dalla tastiera di un PIN per poter completare la transazione

# Software di sicurezza

## Firewall

Un **firewall** è un componente cruciale nelle strategie di sicurezza dei computer e delle reti. Tipicamente, un firewall viene posizionato tra una rete locale e Internet, stabilendo un **collegamento controllato** e agendo come un **muro o perimetro di sicurezza esterno**. Il suo scopo principale è proteggere la rete locale dagli attacchi basati sulla rete e fornire un singolo punto di arresto dove è possibile impostare sicurezza e controllo. I firewall possono anche essere installati all'interno di una rete aziendale per separare le sue porzioni.

### Obiettivi di progettazione di un firewall:

- **Tutto il traffico** (sia dall'interno all'esterno che viceversa) deve passare attraverso il firewall. Ciò si ottiene bloccando fisicamente tutti gli accessi alla rete locale, eccetto quelli che passano per il firewall.
- Solo il **traffico autorizzato**, come definito dalla politica di sicurezza locale, deve essere permesso di attraversare il firewall.
- Il firewall stesso deve essere **immune alla penetrazione**. Questo implica l'uso di un sistema rafforzato con un sistema operativo (OS) sicuro, spesso richiesto in applicazioni governative.

I firewall utilizzano quattro tecniche principali per controllare gli accessi:

- **Controllo dei servizi:** Determina i tipi di servizi Internet che possono essere acceduti, sia in entrata che in uscita. Può filtrare il traffico in base all'indirizzo IP, al protocollo o al numero di porta. Può anche fornire software proxy o ospitare il software del server da proteggere.
- **Controllo della direzione:** Stabilisce la direzione in cui le richieste di servizi specifiche possono essere avviate e consentite attraverso il firewall.
- **Controllo degli utenti:** Regola l'accesso a un servizio in base all'utente che tenta di accedervi. Questa funzione si applica tipicamente agli utenti all'interno del perimetro del firewall (utenti locali) e, tramite tecnologie di autenticazione sicura (come IPsec), anche al traffico in entrata da utenti esterni.
- **Controllo del comportamento:** Controlla come vengono utilizzati servizi specifici. Ad esempio, può filtrare le e-mail per eliminare lo spam o limitare l'accesso esterno solo a una porzione di informazioni su un server web locale.

### **Capacità:**

- Definiscono un **unico punto di arresto** che tiene gli utenti non autorizzati all'esterno della rete protetta e impedisce a servizi potenzialmente vulnerabili di accedere o lasciare la rete.
- Forniscono una posizione per il **monitoraggio degli eventi** legati alla sicurezza, con funzionalità di monitoraggio e gestione degli allarmi.
- Servono come **piattaforma per funzioni di rete** non legate alla sicurezza, come il NAT (Network Address Translator) o la gestione della rete.
- Possono fungere da piattaforma per implementare **reti private virtuali (VPN)**.

### **Limiti:**

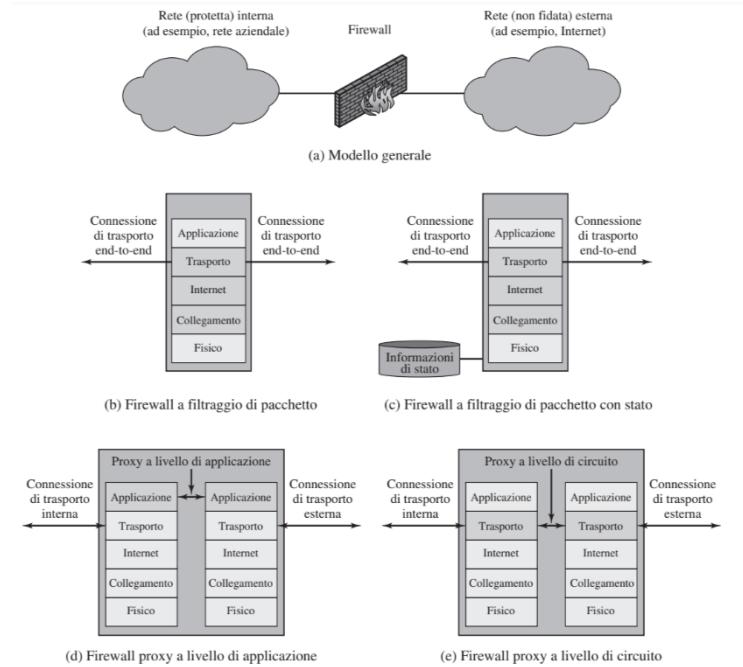
- Non possono proteggere da **attacchi che bypassano il firewall**, ad esempio, sistemi interni che si connettono direttamente a un ISP o LAN interne con modem pool.
- Non possono proteggere completamente dalle **minacce interne**, come dipendenti scontenti o che cooperano involontariamente con un attaccante esterno.
- Una LAN wireless non adeguatamente protetta può essere raggiunta dall'esterno dell'organizzazione, bypassando i firewall interni.
- Dispositivi portatili (laptop, smartphone, USB drive) possono essere infettati all'esterno e poi collegati e usati internamente, reintroducendo malware.

**Tipi di firewall** I firewall possono agire come filtri di pacchetto, operando come filtri positivi (permettendo solo ciò che è specificamente consentito) o negativi (bloccando ciò che è specificamente proibito).

### **Firewall a filtraggio di pacchetto:**

1. Applica un insieme di regole a ogni pacchetto IP in ingresso e in uscita, per inoltrarlo o scartarlo.
2. Le regole si basano su informazioni nell'intestazione del pacchetto, come indirizzo IP sorgente/destinazione, porta di trasporto sorgente/destinazione, campo protocollo dell'intestazione IP e interfaccia di provenienza/destinazione del pacchetto.
3. **Politiche di default:** *Default = discard* (ciò che non è espressamente permesso è proibito - più conservativa) o *Default = forward* (ciò che non è espressamente proibito è permesso - meno sicura ma più facile da usare).
4. **Debolezze:** Non esaminano i dati degli strati superiori, quindi non possono prevenire attacchi che sfruttano vulnerabilità specifiche delle applicazioni. La funzionalità di logging è limitata. Non supportano schemi avanzati di autenticazione utente. Sono vulnerabili ad attacchi che sfruttano problemi delle specifiche TCP/IP (es. spoofing degli indirizzi a

livello di rete). Sono suscettibili a violazioni di sicurezza dovute a configurazioni errate.



### Attacchi e contromisure:

- **Spoofing dell'indirizzo IP:** Intruso invia pacchetti dall'esterno con un indirizzo sorgente interno. Contromisura: scartare pacchetti con indirizzo sorgente interno se provenienti da un'interfaccia esterna.
- **Attacchi di tipo "source routing":** Sorgente specifica il percorso del pacchetto. Contromisura: scartare pacchetti che usano questa opzione.
- **Attacchi con piccoli frammenti:** Attaccante usa frammenti IP molto piccoli per eludere le regole di filtraggio basate sull'header TCP. Contromisura: applicare una regola che impone una dimensione minima dell'header di trasporto nel primo frammento e scartare i frammenti successivi se il primo viene rifiutato.

### Firewall a filtraggio di pacchetto con stato:

- Estende il filtraggio di pacchetto mantenendo un **elenco delle connessioni TCP in uscita**. Permette il traffico in ingresso verso porte con alta numerazione solo per i pacchetti che corrispondono a una connessione stabilita.
- Possono tracciare i numeri di sequenza TCP per prevenire attacchi di dirottamento delle sessioni e controllare limitate quantità di dati a livello applicativo per protocolli noti.

### Gateway a livello di applicazione (o proxy di applicazione):

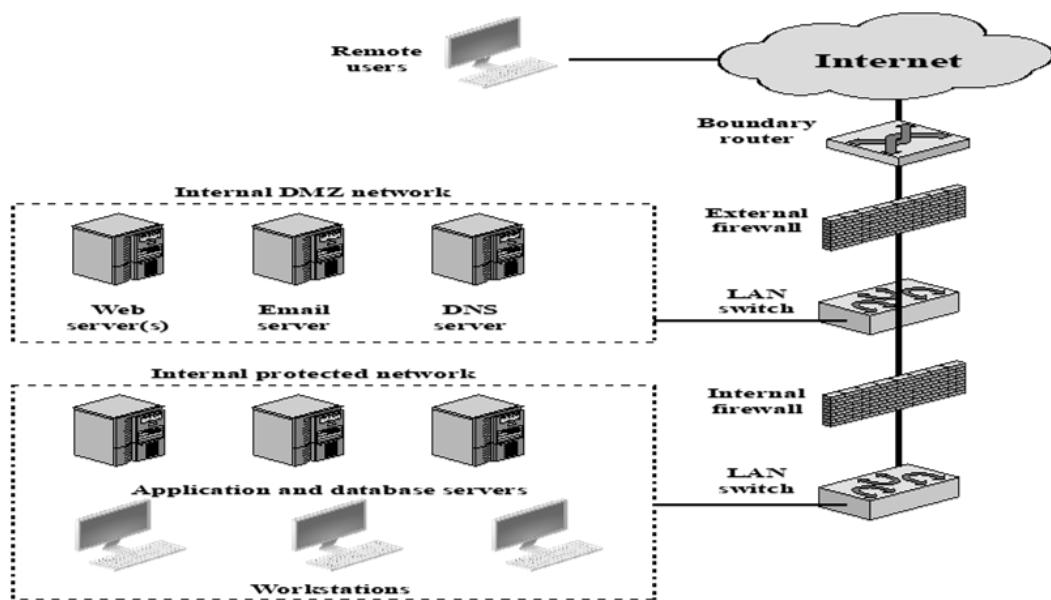
- Agisce come un **relay di traffico a livello applicativo**. L'utente contatta il gateway, che a sua volta contatta l'host remoto. Non

permette una connessione TCP end-to-end diretta tra utente e server.

- Più sicuri dei filtri di pacchetto perché esaminano solo le applicazioni consentite e facilitano il logging.
- **Svantaggio:** Sovraccarico di elaborazione a causa delle due connessioni separate e del relay del traffico.

### Gateway a livello di circuito (o proxy a livello di circuito):

- Simile al gateway a livello di applicazione, ma non esamina il contenuto dei segmenti TCP una volta che le due connessioni (una interna e una esterna) sono stabilite.
- La funzione di sicurezza consiste nel determinare quali connessioni sono permesse.
- Tipico in situazioni in cui l'amministratore si fida degli utenti interni, gestendo il sovraccarico di analisi solo per le connessioni in ingresso.



### Reti DMZ (Demilitarized Zone):

- Una DMZ è una **regione di rete tra un firewall esterno e uno o più firewall interni**.
- I sistemi accessibili dall'esterno che necessitano di protezione (es. server web, e-mail, DNS aziendali) sono tipicamente situati nelle reti DMZ.
- Il firewall esterno controlla e protegge i sistemi DMZ, fornendo anche una protezione di base per il resto della rete interna.
- I **firewall interni** aggiungono filtri più stringenti, proteggono il resto della rete da attacchi provenienti dalla DMZ (es. da malware), e possono proteggere diverse porzioni della rete interna tra loro.

### Firewall nel contesto di altri sistemi di sicurezza:

- I firewall sono un complemento importante ai servizi di sicurezza basati su host, come i sistemi di rilevamento delle intrusioni (IDS).
- Possono bloccare il flusso di traffico sospetto, impedendo che raggiunga i sistemi target.
- Nelle reti wireless, i firewall possono essere utilizzati per limitare la visibilità dei dati e l'accesso alle applicazioni per tutti i dispositivi mobili.
- In ambienti virtualizzati o cloud, i **firewall virtuali** proteggono il traffico di rete tra i sistemi senza doverlo instradare su una rete fisica separata.

Possono essere implementati come:

- **VM Bastion Host:** Una VM specifica funge da bastion host, monitorando e instradando il traffico tra sottoreti virtuali, replicando le funzionalità di un bastion host fisico.
- **VM Host-Based Firewall:** Il sistema operativo guest all'interno della VM include un firewall basato su host per proteggere la VM stessa.
- **Firewall hypervisor:** Le funzionalità del firewall sono fornite direttamente dall'hypervisor, permettendo un monitoraggio e filtraggio più efficiente e sicuro.

## NIDS

Un **NIDS (Network-Based Intrusion Detection System)** è un tipo di sistema di rilevamento delle intrusioni che monitora il traffico in specifici segmenti di rete o dispositivi, analizzando i protocolli a livello di rete, trasporto e applicazione per identificare attività sospette.

Un NIDS funziona osservando i pacchetti che transitano sulla rete attraverso un sensore. Questi pacchetti sono considerati di interesse se corrispondono a una "signature".

Un NIDS solitamente realizza il suo compito mettendo la scheda di interfaccia di rete in **modalità promiscua** per catturare tutto il traffico di rete che attraversa il suo segmento. Tuttavia, il traffico su altri segmenti o su altri mezzi di comunicazione (come le linee telefoniche) non può essere monitorato da un singolo NIDS. Gli IDS in generale, inclusi i NIDS, sono composti da tre componenti logiche:

1. **sensori**, raccolgono dati;
2. **analizzatori**, determinano se si è verificata un'intrusione;
3. **interfaccia utente**, per osservare l'output o controllare il sistema

## Cosa registrano

- Timestamp
- Connessione o session ID
- Tipo di evento o avviso

- Protocolli di rete, trasporto e livello applicativo
- Origine e destinazione degli indirizzi IP
- Porte di origine e destinazione TCP o UDP
- Numero di byte trasmessi tramite le connessioni
- Dati dei payload decodificati, come richieste e risposte dell'applicazione
- Informazioni relative allo stato delle connessioni

**Tipi di Signature utilizzati da un NIDS** I NIDS identificano le attività sospette utilizzando principalmente tre tipi di signature (firme):

1. **Signature basate su stringhe:** cercano una specifica stringa di testo che indichi un possibile attacco. Ad esempio, per UNIX, una stringa come "cat "+">./rhosts" potrebbe rendere un sistema vulnerabile agli attacchi di rete. Per ridurre i falsi positivi, possono essere utilizzate signature composte, combinando più stringhe, come "cgi-bin" AND "aglimpse" AND "IFS" per attacchi a server Web.
2. **Signature basate sulle porte:** cercano tentativi di connessione a porte ben note e frequentemente attaccate, come Telnet (porta TCP 23), FTP (porta TCP 21/20), SUNRPC (porta TCP/UDP 111) e IMAP (porta TCP 143). Se queste porte non sono utilizzate dal sito, i pacchetti in ingresso su di esse sono considerati sospetti.
3. **Signature basate su header:** osservano combinazioni pericolose o illogiche nelle intestazioni dei pacchetti. Un esempio notevole è un pacchetto TCP con entrambi i flag SYN e FIN impostati, il che implica un tentativo di avviare e interrompere una connessione contemporaneamente.

**Principi Base del Rilevamento delle Intrusioni** Il rilevamento delle intrusioni, incluso quello basato sulla rete, è motivato da diverse considerazioni:

- a. Un rilevamento tempestivo può consentire di identificare e rimuovere l'intruso prima che si verifichino danni significativi o compromissioni dei dati.
- b. Un IDS efficace può agire come deterrente, prevenendo intrusioni.
- c. Il rilevamento delle intrusioni permette di raccogliere informazioni sulle tecniche utilizzate dagli intrusi, che possono poi essere impiegate per rafforzare le misure di prevenzione. Il rilevamento delle intrusioni si basa sull'assunto che il comportamento di un intruso sia quantificabilmente diverso da quello di un utente legittimo, sebbene ci possano essere sovrapposizioni.

**Approcci al Rilevamento delle Intrusioni** Esistono due approcci generali per il rilevamento delle intrusioni:

1. **Rilevamento di usi non autorizzati (misuse detection)**: si basa su regole che specificano eventi o proprietà di sistema sintomatiche di incidenti di sicurezza. Utilizza algoritmi di pattern matching su database di firme di attacco (signature). Il vantaggio è l'accuratezza e pochi falsi allarmi, ma non può rilevare attacchi nuovi o sconosciuti.
2. **Rilevamento delle anomalie (anomaly detection)**: cerca attività che si discostano dal comportamento normale delle entità e risorse del sistema. Il vantaggio è la capacità di rilevare attacchi sconosciuti, ma presenta un compromesso significativo tra falsi positivi (utenti legittimi identificati come intrusi) e falsi negativi (intrusi non identificati).

**Disposizione (Deployment) di un NIDS** I sensori NIDS devono essere posizionati strategicamente in punti chiave della rete per raccogliere passivamente i dati di traffico. Esistono quattro tipi principali di posizioni per i sensori:

1. **All'esterno del firewall principale**: utile per valutare il livello di minaccia generale per la rete aziendale.
2. **Nella rete DMZ (Demilitarized Zone)**: all'interno del firewall principale ma all'esterno dei firewall interni. Questa posizione permette di monitorare i tentativi di penetrazione che mirano a servizi aperti agli estranei (come Web o e-mail).
3. **Dietro i firewall interni, sulle reti dorsali**: per monitorare le principali dorsali di rete, come quelle che supportano i server interni e le risorse dei database.
4. **Dietro i firewall interni, sulle LAN degli utenti**: per monitorare le LAN che supportano le workstation degli utenti e i server dipartimentali. Le posizioni 3 e 4 consentono di rilevare attacchi più specifici a segmenti di rete e quelli che originano dall'interno dell'organizzazione.

**Relazione con altri sistemi di sicurezza** I NIDS sono un complemento dei firewall. I firewall possono bloccare il flusso di traffico sospetto, impedendo che raggiunga i sistemi target. Inoltre, i NIDS si distinguono dagli **IDS host-based**, i quali monitorano le caratteristiche di un singolo host e gli eventi al suo interno, potendo determinare con precisione quali processi e account utente sono coinvolti in un attacco. A differenza dei NIDS, gli IDS host-based possono anche vedere più facilmente l'esito di un tentativo di attacco, monitorando direttamente file e processi di sistema.

## AntiSpyware

L'antispyware è una categoria di software progettata specificamente per il **rilevamento e la rimozione dello spyware**. L'obiettivo principale

dell'antispyware è raccogliere informazioni da un computer e trasmetterle a un altro sistema monitorando sequenze di caratteri digitati su tastiera (keylogging), dati dello schermo, traffico di rete o scansionando file sul sistema alla ricerca di informazioni sensibili. Il software antispyware aiuta a contrastare queste attività malevoli.

## Antivirus

L'antivirus, più ampiamente definito come **contromisure per i software malevoli (malware)**, rappresenta una difesa fondamentale per i sistemi informatici e le reti.

L'obiettivo delle contromisure, inclusi i meccanismi antivirus, è la **prevenzione** del malware dal sistema o il blocco della sua capacità di modificarlo. Tuttavia, l'adozione di adeguate contromisure può "ridurre significativamente il numero di attacchi malware con successo".

Quattro fattori principali di prevenzione:

- **Policy** (politiche).
- **Sensibilizzazione** (awareness).
- **Mitigazione delle vulnerabilità**: Mantenere tutti i sistemi aggiornati con le patch, riducendo le vulnerabilità sfruttabili.
- **Mitigazione delle minacce**: Impostare adeguati controlli di accesso ad applicazioni e dati per limitare l'infezione o la corruzione da parte del malware.

Se le misure di prevenzione falliscono, si ricorre a meccanismi tecnici di mitigazione delle minacce che si concentrano su:

- **Rilevamento**: Accertare l'esistenza del malware una volta che l'infezione si è verificata.
- **Identificazione**: Individuare lo specifico malware responsabile dell'infezione.
- **Rimozione**: Eliminare tutte le tracce del malware dai sistemi infetti. Se identificazione o rimozione non sono possibili, si deve ricorrere all'eliminazione dei file infetti e al ricaricamento di una versione di backup pulita, o in casi gravi, alla cancellazione completa dello spazio di archiviazione e alla ricostruzione del sistema.

## Posizionamento e Tipi di Meccanismi Antivirus

I meccanismi di rilevamento del malware possono essere collocati in diverse posizioni.

**Scanner host-based e anti-virus basati su firme:** Operano direttamente su ogni sistema terminale (es. personal computer), fornendo un accesso ampio alle informazioni sul comportamento del malware.

1. **Prima generazione (scanner semplici):** Basati su "firme" (pattern di bit costanti) di malware noti o su cambiamenti di lunghezza dei programmi.
2. **Seconda generazione (scanner euristici):** Usano criteri euristici per individuare istanze di malware, cercando frammenti di codice associati a essi o utilizzando **controlli di integrità** (checksum o funzioni hash cifrate).
3. **Terza generazione (trap di attività):** Programmi residenti in memoria che identificano il malware in base alle sue azioni (behavior-blocking), anziché alla sua struttura, utilizzando **analisi dinamica**. Possono bloccare azioni dannose in tempo reale, indipendentemente dall'offuscamento del codice.
4. **Quarta generazione (protezione completa):** Pacchetti integrati che combinano scanner, trap di attività e funzionalità di **controllo degli accessi** per limitare la penetrazione e la propagazione del malware.

**Analisi Sandbox:** Consiste nell'esecuzione di codici potenzialmente dannosi in un ambiente emulato o su una macchina virtuale per monitorarne il comportamento. Questo aiuta a rilevare malware complessi, criptati, polimorfi o metamorfici e a sviluppare nuove firme. Tuttavia, il malware può rilevare l'ambiente sandbox e sopprimere il suo comportamento malevolo o usare lunghi periodi di sospensione per eludere il rilevamento.

**Scansione perimetrale:** Integrata nei firewall e nei sistemi di rilevamento delle intrusioni (IDS) delle organizzazioni. Ciò consente di accedere al malware in transito su una connessione di rete e di bloccare il traffico sospetto.

- **Monitor di ingresso:** Posti al confine tra la rete aziendale e Internet, utilizzano anomalie, firme ed euristiche per rilevare il traffico malware.
- **Monitor di uscita:** Collocati ai punti di uscita delle LAN o al confine, rilevano la sorgente di un attacco monitorando il traffico in uscita per attività di scansione o traffico e-mail anomalo (es. worm di massa). Possono implementare contromisure per la prevenzione della perdita di dati (data exfiltration).

Il monitoraggio perimetrale contribuisce anche a rilevare l'attività delle botnet, cercando di rilevarle e disabilitarle durante la fase di costruzione.

Il **Digital Immune System** è progettato per **monitorare costantemente un ambiente IT**, individuare comportamenti anomali, **identificare minacce in tempo reale** e rispondere in maniera automatica, rapida e adattiva.

A differenza dei tradizionali sistemi di sicurezza, che spesso si basano su regole statiche e definizioni di virus note, un sistema immunitario digitale utilizza tecnologie avanzate come **machine learning**, **intelligenza artificiale** e **analisi comportamentale** per imparare cosa è “normale” all’interno dell’ambiente operativo. Grazie a questo apprendimento continuo, riesce a individuare minacce **mai viste prima**, comprese quelle sofisticate che sfuggono ai controlli convenzionali.

Una delle caratteristiche principali di questo approccio è la **capacità di reazione autonoma**. Quando viene rilevata un’anomalia, il sistema può ad esempio isolare una macchina sospetta dalla rete, bloccare un utente compromesso, o annullare modifiche malintenzionate — tutto senza richiedere un intervento umano immediato. Questo riduce drasticamente i tempi di risposta e può impedire che un attacco si diffonda all’interno dell’organizzazione.

Il sistema immunitario digitale non è un singolo strumento, ma piuttosto una **combinazione integrata** di più componenti di sicurezza: firewall, sistemi di rilevamento e prevenzione delle intrusioni (IDS/IPS), antivirus, soluzioni per il monitoraggio degli endpoint (EDR), strumenti di orchestrazione e risposta agli incidenti (SOAR), e sistemi di gestione degli eventi (SIEM). Tutti questi elementi lavorano insieme, come un’unica “rete difensiva intelligente”, scambiandosi informazioni in tempo reale.

Un esempio concreto del suo funzionamento potrebbe essere il seguente: un programma inizia a comportarsi in modo anomalo all’interno di una rete aziendale. Il sistema immunitario digitale rileva questa deviazione rispetto al comportamento usuale, isola automaticamente la macchina compromessa, blocca le azioni dannose e avvia un’analisi dell’incidente. Infine, aggiorna le proprie regole interne per riconoscere comportamenti simili in futuro. In questo modo, il sistema non solo reagisce, ma **impara e si adatta costantemente**.

Oltre alla protezione tecnica, un digital immune system comprende anche componenti organizzative: politiche di sicurezza, formazione del personale, gestione delle vulnerabilità e preparazione alla risposta agli incidenti. Questo approccio integrato è fondamentale per affrontare un panorama delle minacce sempre più complesso e in rapido cambiamento.

## STRATEGIE DI SICUREZZA

Una **strategia di sicurezza informatica complessiva** è fondamentale per la protezione delle **risorse informatiche** da minacce e attacchi. Le strategie mirano a soddisfare gli obiettivi chiave della sicurezza informatica, spesso riassunti nella **triade CIA** e anche dell'**Autenticità e Responsabilità**.

Una strategia di sicurezza coinvolge tre aspetti principali:

- **Specifiche/politica:** Definisce ciò che lo schema di sicurezza dovrebbe fare.
- **Attuazione/meccanismi:** Descrive come lo schema di sicurezza dovrebbe essere realizzato.
- **Correttezza/garanzia:** Valuta se lo schema di sicurezza funziona effettivamente come previsto.

### Politica di sicurezza

Il primo passo per concepire servizi e meccanismi di sicurezza è lo sviluppo di una **politica di sicurezza**. Questa è una dichiarazione formale di regole e pratiche che specificano o regolano come un sistema o un'organizzazione debba fornire servizi di sicurezza per proteggere le risorse sensibili e critiche.

Nel definire una politica di sicurezza, un responsabile della sicurezza deve considerare:

- Il **valore delle risorse** da proteggere.
- Le **vulnerabilità** del sistema.
- Le **minacce potenziali** e la probabilità degli attacchi.

Inoltre, devono essere bilanciati i seguenti compromessi:

- **Facilità d'uso rispetto alla sicurezza:** Quasi tutte le misure di sicurezza comportano uno svantaggio in termini di usabilità. Ad esempio, i meccanismi di controllo degli accessi richiedono la memorizzazione di password, e i firewall possono ridurre la capacità di trasmissione.
- **Costo della sicurezza rispetto al costo per malfunzionamento e ripristino:** I costi diretti di implementazione e manutenzione delle misure di sicurezza devono essere bilanciati con i costi di un fallimento della sicurezza e del ripristino.

La politica di sicurezza è, in ultima analisi, una decisione aziendale, che può essere influenzata da requisiti legali.

### Attuazione della sicurezza

L'implementazione della sicurezza implica quattro linee d'azione complementari:

- **Prevenzione:** L'obiettivo ideale è che nessun attacco abbia successo. Ad esempio, l'uso di algoritmi di cifratura robusti e la protezione delle chiavi

possono prevenire attacchi alla confidenzialità dei dati trasmessi. Per i malware, i fattori di prevenzione includono politiche, sensibilizzazione, mitigazione delle vulnerabilità e mitigazione delle minacce.

- **Rilevazione:** Quando la protezione assoluta non è realizzabile, è cruciale rilevare gli attacchi. Esempi includono i sistemi di rilevamento delle intrusioni (IDS) o il rilevamento di attacchi Denial of Service (DoS). La rilevazione può anche fungere da deterrente.
- **Risposta:** Se un attacco in corso viene rilevato (es. un attacco DoS), il sistema deve essere in grado di rispondere per fermare l'attacco e prevenire ulteriori danni.
- **Ripristino:** Prevede l'utilizzo di meccanismi per riportare il sistema o i dati a uno stato corretto dopo una violazione della sicurezza, ad esempio tramite sistemi di backup.

## Garanzia e valutazione

Coloro che si affidano ai servizi e ai meccanismi di sicurezza informatica desiderano avere la **fiducia** che le misure di sicurezza implementate funzionino come previsto.

- **Garanzia (Assurance):** Si riferisce al grado di fiducia che un sistema informativo operi in conformità con la politica di sicurezza stabilita, sia nella fase di progettazione che in quella di implementazione. La garanzia viene espressa come un livello di fiducia, non come una prova formale.
- **Valutazione (Evaluation):** È il processo di esame di un prodotto o sistema informatico rispetto a specifici criteri, e può includere test e tecniche analitiche o matematiche formali.

La sicurezza informatica è un campo complesso, e la sua strategia deve tenere conto di diverse sfide:

- La complessità dei meccanismi di sicurezza e la loro natura spesso controidintuitiva.
- La necessità di gestire informazioni segrete, come le chiavi di cifratura, e i problemi di creazione, distribuzione e protezione di queste informazioni.
- La lotta costante tra attaccanti e progettisti: l'attaccante deve trovare una singola debolezza, mentre il progettista deve eliminarle tutte.
- La percezione da parte di utenti e manager che gli investimenti in sicurezza portino pochi benefici fino a quando non si verifica un incidente.
- La tendenza a integrare la sicurezza solo dopo il completamento del progetto, piuttosto che renderla parte integrante sin dall'inizio.
- L'ostacolo che una sicurezza elevata può rappresentare per l'efficienza e la facilità d'uso di un sistema.