SUID

È un permesso speciale che si attribuisce ai file. SUID è definito come dare i permessi temporanei di un utente, di eseguire un programma / file con i permessi del proprietario del file piuttosto che l'utente che lo gestisce.

Esempio di Trojan Horse

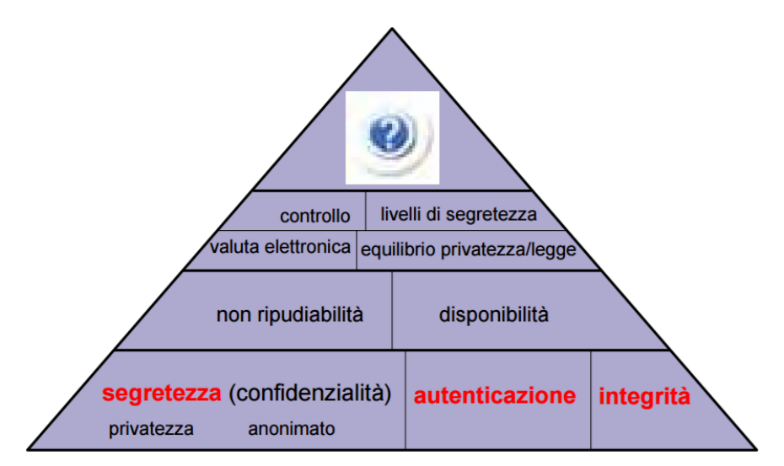
Immagine che contiene testo

Descrizione generata automaticamente

Definizione di **sicurezza**: prevenzione attacchi + corretto funzionamento del sistema.

Ad esempio, un computer spento è sicuro, ma non rispetta la seconda parte, ovvero non sta funzionando.

Proprietà della sicurezza



Il **non-ripudio** indica l’impossibilità di dire falsità, di negare la propria partecipazione in una sessione di comunicazione; qualcosa del tipo: posta elettronica certificata.

La **disponibilità** ha a che fare con il fatto che se abbiamo un conto bancario online e lo troviamo down, non piace la cosa. Non è chiaro che nesso ci sia tra la disponibilità e la sicurezza in generale. Non è perché il sito è down è sicuro. Magari è down per manutenzione. Però la disponibilità del servizio è considerata, per ragioni socio-tecnologiche, un fattore di affidabilità.

**Controllo**: si riferisce al controllo di accesso.

**Livelli di segretezza**: per i sistemi mandatori, sistemi militari

**Equilibrio privatezza/legge**: privatezza dell’individuo verso ciò che la legge vuole fare

Segretezza: L’informazione non sia rilasciata ad entità non autorizzata a conoscerla. È segreto solo se lo conoscono le persone giuste. Per rendere qualcosa “segreto” usiamo la crittografia o la stenografia per esempio.

Autenticazione: Le entità devono essere chi dichiarano di essere.

Integrità: L’informazione non sia modificata da entità non autorizzate. Integrità delle informazioni: il sistema deve impedire la alterazione diretta o indiretta delle informazioni, sia da parte di utenti e processi non autorizzati, che a seguito di eventi accidentali. Anche la perdita di dati (per esempio a seguito di cancellazione o danneggiamento) viene considerata come alterazione. Un soggetto non autorizzato entra in possesso di una componente del sistema, la modifica e la introduce di nuovo nel sistema. Questo è un attacco all’integrità.

Privatezza: Diritto di un’entità di rilasciare o meno i propri dati personali ad altre entità. La privatezza sembra simile alla segretezza, ma a differenza di quest’ultima, **la privatezza parla di diritto**. Questa definizione è basata sulle leggi Europee (in America questo diritto è molto discusso). Quindi la privatezza è un diritto di segretezza dei propri **dati personali**. La privatezza, a differenza della segretezza, tratta un ben preciso tipo di dati, i dati personali, dette informazioni personali statiche.

Anonimato: Diritto dell’iniziatore di una transazione di rilasciare o meno la propria identità ad altre entità. Un **proxy** è un programma che si interpone tra un client ed un server facendo da tramite o interfaccia tra i due host, ovvero inoltrando le richieste e le risposte dall'uno all'altro. Il client si collega al proxy invece che al server, e gli invia delle richieste. Il proxy a sua volta si collega al server e inoltra la richiesta del client, riceve la risposta e la inoltra al client. A differenza di bridge e router, che lavorano ad un livello ISO/OSI più basso (in quanto sfruttano i NAT), i proxy nella maggior parte dei casi lavorano a livello applicativo; di conseguenza, un programma proxy gestisce un numero limitato di protocolli applicativi.

Non-ripudio: L’entità non possa negare la propria partecipazione ad una transazione con uno specifico ruolo. Un esempio concreto è la PEC, dove, essendoci una ricevuta, non si può negare di aver partecipato alla transazione.

Si può essere autenticato e anonimo allo stesso tempo? Consideriamo l’esame scritto di sicurezza. È stato ideato un protocollo detto **WATA**. Quest’ultimo permetteva al professore di correggere i compiti scritti senza sapere di chi fosse quel compito. Quindi garantiva allo studente di essere autenticati e anonimi allo “stesso tempo”. Con questo procedimento non è impossibile avere utenti autenticati e anonimati “contemporaneamente”. Ma attenzione, non accade nello stesso istante, cioè prima l’utente si autentica, ma solo in un secondo momento diventa anonimato. Questo passaggio avveniva nel momento in cui viene rimosso il token dal compito.

Disponibilità: Il sistema sia operante e funzionante in ogni momento.

Abbiamo il problema tra la disponibilità e la negazione del servizio. Un client potrebbe mandare tante richieste a un servizio online (**flooding** di richieste). Un protocollo (per cercare di risolvere il problema descritto prima), si chiama cookie transformation, riguarda di complicare l’accesso al sistema impegnando computazionalmente il chiamante. Questo protocollo funziona così: noi facciamo una domanda, il destinatario prima di rispondere ci fa un’altra domanda. Quindi se il chiamante facesse un ciclo infinito di flooding, allora anche lui deve impegnarsi computazionalmente. Rispondere con un'altra domanda è computazionalmente meno impegnativo di rispondere a una risposta voluta.

Controllo di accesso: Ciascun utente abbia accesso a tutte e sole le risorse o i servizi per i quali è autorizzato. Ecco alcune misure per il controllo di accesso:

**Autenticazione dell’utente**: dobbiamo controllare ogni utente; che cosa possa fare, che cosa deve fare. Quindi prima di tutto dobbiamo controllare che tipo di utente sia.

**Politiche di sicurezza.**

Il problema delle politiche di sicurezza sono le regole che dobbiamo dare. Senza di esse non ci possiamo aspettare che il sistema sia sicuro.**Immagine che contiene testo

Descrizione generata automaticamente**

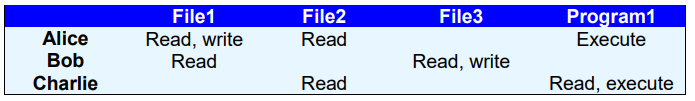
**MAC**

Il MAC (Mandatory Access Control) realizza politiche basate sull’obbligo. E’ usato in ambito militare. È un meccanismo di sicurezza che limita il livello di controllo che gli utenti (soggetti) hanno sugli oggetti che essi creano. A differenza di un DAC di attuazione, in cui gli utenti hanno il pieno controllo sui propri file, directory, ecc., MAC aggiunge etichette aggiuntive, o categorie, a tutti gli oggetti del file system. Utenti e processi devono avere l'accesso appropriato a queste categorie prima di poter interagire con questi oggetti.

**RBAC**

Role-Based Access Control realizza politiche non mandatorie, ovvero basate su permessi associati a ciascun ruolo e modificabili. Vale solo la “legge” del Vietato e Permesso, scomparso l’obbligatorio.

Per mettere in pratica le politiche possiamo usare le ACM, ovvero le matrici di controllo di accesso. Qui possiamo specificare chi può fare cosa.

****

Le colonne rappresentano ACL (Access Control List), mentre le righe rappresentano CaL (Capability List).

**Modelli di Attaccanti**

Un modello di attaccante specifica le capacità offensive di un preciso attaccante.

Fissate un insieme di capacità offensive, gli studi di sicurezza li possiamo fare in prelazione a quelle capacità offensive. Un modello di attaccante notevole si chiama **Dolev-Yao.** Questo modello sarà un modello “pessimo”, ovvero il peggiore possibile in termini di capacità offensiva (quindi è un modello massimamente offensivo). Se abbiamo dimostrato che il sistema è sicuro nei confronti dell’attaccante massimamente offensivo, allora a maggior ragione sarà sicuro con un attaccante meno offensivo.

**Modello di attaccante Dolev-Yao (DY)**

L’attaccante è unico e superpotente, ovvero controlla l’intera rete, ma non può violare la crittografia.

**Modello di attaccante General Attacker (GA)**

Chiunque può essere attaccante senza interesse a colludere gli altri, al peggio con capacità di totale controllo della rete, ma senza violare la crittografia.

**Autenticazione**

L’autenticazione si fa per mezzo di qualcosa che ci caratterizzi, perché per la definizione di autenticazione ognuno è quello che dice di essere. Quindi se ci arriva una e-mail da Giampaolo Bella questa email tipicamente non sarà autenticata, ma se mi arriva una email della caratterizzazione di Giampaolo Bella allora discerneremo che è veramente Giampaolo Bella. I modi per discernere sono:

* **Conoscenza**
* **Possesso**
* **Informazioni Biometriche**

È un concetto semplice per la nostra epoca, perché ad esempio se il bancomat non avesse il pin, quindi sarebbe solo autenticazione basata sul possesso, sarebbe poco robusta. Quindi dobbiamo fare autenticazione basata sul possesso, (con la carta) e sulla conoscenza (ovvero avere il pin). Quindi sia se considero solo conoscenza o solo possesso, queste tecniche sono “deboli” da sole, ma si possono potenziare l’una con l’altra.

**Cenni di crittografia**

**Immagine che contiene testo

Descrizione generata automaticamente**

In caso di decryption, se la chiave non coincide con l’originale, ma usiamo una chiave k’, allora la funzione di decrypting ci restituirà un messaggio m’. Più la chiave è simile a k, più il messaggio lo sarà.

Nei sistemi di encryption TUTTI conoscono gli algoritmi, ma solo il RICEVENTE deve conoscere la chiave. È lei infatti che rimane segreta.

**Crittografia simmetrica**

Fissato un agente A, esso sia munito di chiave simmetrica, indicata come **Ka**è detta **chiave a lungo termine** perché il suo intervallo di validità è molto lungo, indipendente dalla specifica sessione di comunicazione. Consideriamo di voler mandare un segreto a un interlocutore che sta in rete e di avere una chiave a lungo termine. Alice codifica un messaggio con Ka e lo manda a Bob. Bob però non può decrittare senza la chiave, ma non posso nemmeno dare a Bob la chiave a lungo termine perché la chiave mi impersonifica. In un qualche modo le chiavi di tutti devono essere note, ma questa cosa non va bene.

**Immagine che contiene testo

Descrizione generata automaticamente**

**Crittografia asimmetrica**

Immagine che contiene testo

Descrizione generata automaticamente

Se Alice ha la chiave k allora la sua inversa è 𝒌-1. Queste chiavi le possiamo generare insieme, ma data una chiave calcolare la sua metà, la sua inversa, sarà un problema computazionalmente intrattabile.

Immagine che contiene testo, calibro, dispositivo

Descrizione generata automaticamente

Come si manda quindi un messaggio con questo sistema? Ogni utente avrà due chiavi, una l’inversa dell’altra. Una viene pubblicata a tutti, ovvero Ka, mentre Ka-1 viene resa privata, solo per Alice. Quindi, se utilizza la chiave privata per criptare, tutti potranno decriptarla. Se utilizza quella pubblica per criptare, nessuno potrà però decriptare il messaggio. Che soluzione adottiamo?

Si utilizza la chiave pubblica del **destinatario** per criptare il messaggio, così che solo il destinatario potrà decriptare il messaggio!

**Limite della crittografia asimmetrica**

In rete, non esistono solamente Alice e Bob, bensì anche altri utenti, ognuno con le sue chiavi. Se Bob ricevesse un messaggio codificato con Kc invece di Kb riceverebbe una stringa a caso, ma non solo, l’utente C saprebbe il messaggio in chiaro! Abbiamo quindi un problema di autenticazione, bisognerebbe sapere se quella chiave appartiene a chi di dovere. Qui entra in gioco la **certificazione**.

**Hash crittografico**

Immagine che contiene testo

Descrizione generata automaticamente

Una funzione hash crittografica è una funzione che prende in ingresso un messaggio e restituisce una stringa di lunghezza fissa. Il messaggio codificato prende il nome di “digest” e varia completamente anche solo cambiando un bit del messaggio.

L’hash **non è** encryption. Una password criptata con una chiave può essere “rotta” e quindi decriptata. Una funzione hash non è invertibile, quindi una password con funzione hash non è rompibile, e questa è un’ottima base a livello di segretezza.

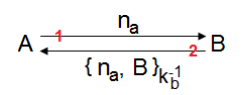
**Salting**

Il salt è un messaggio random aggiunto ad una password per proteggerla da attacchi dizionario prima di applicare l’hash. Esso era storicamente di 12 bit.

**Freshness**

La freshness è una meta-proprietà di sicurezza. Quando sono in una sessione, devo essere autenticato costantemente. Chi mi assicura che dietro lo schermo c’è lo stesso tizio autenticato 10 minuti prima? Serve un controllo di autenticazione costante. Se qualcuno logga e poi se ne va, e arriva un tizio dopo 10 minuti e prova a loggare, esso dovrebbe avere davanti di nuovo la schermata di login. Se invece rimane collegato c’è un problema di freshness. Essa possiamo applicarla in due modi: **timestamp** e **nonce**. Il nonce è stato creato con l’idea di creare un numero casuale e usarlo solo una volta. La OTP (one time password) è un esempio.

Per far sì che A autentichi B, utilizziamo la nonce. Dobbiamo stare attenti però che nonce la crei chi vuole ricevere garanze di freshness, quindi A. Crea quindi na e lo manda a B. B manda indietro il numero random di A concatenato alla sua autenticazione, il tutto criptato con la sua chiave privata, così che A, avendo la chiave pubblica di B, può decrittare il tutto ed essere sicuro che ci sia B dietro perché ha utilizzato la sua chiave privata che nessuno conosce.



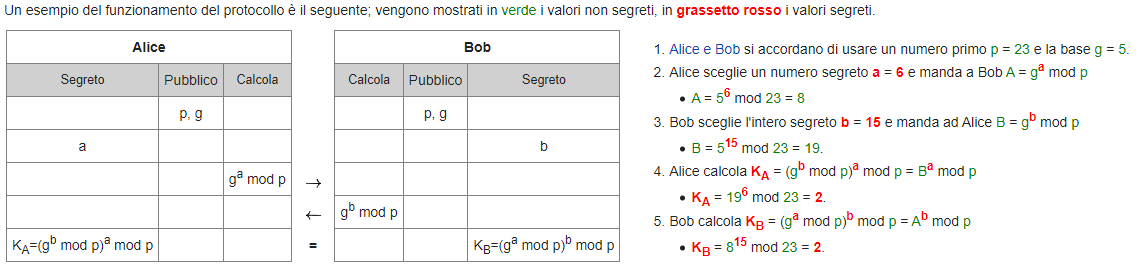
**Firma digitale**

La firma digitale riesce ad unire l’autenticazione all’integrità del messaggio. Questo avviene perché al messaggio viene concatenato l’hash del messaggio stesso criptato con la chiave privata del mittente. Se il destinatario riesce a decrittare l’hash, significa appunto che il messaggio viene da A. Per dimostrare poi che il messaggio non sia stato alterato prova ad usare l’hash sul messaggio e vedere se corrisponde allo stesso ricevuto nella concatenazione!



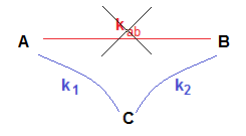
**Protocollo Diffie-Hellmann**

Esso è un protocollo usato sul lato simmetrico e serve per generare la chiave di sessione e condividerla segretamente. Esso sfrutta le proprietà delle potenze, e, a causa dei calcoli fatti, non può essere calcolato il logaritmo discreto.



I due comunicanti scelgono un numero primo **p** (solitamente molto alto) e una base **g**. Poi, entrambi scelgono un numero casuale segreto **a** e **b** e lo usano per calcolare due numeri pubblici **A** e **B**. Una volta mandati reciprocamente questi due numeri essi calcolano due ultimi numeri che, tramite un’operazione di potenza, darà un risultato uguale per entrambi. Quel risultato sarà la chiave di sessione per criptare.

C’è però un problema: non abbiamo autenticazione. Nessuno dei due interlocutori sa per certo se quello dall’altro lato sia chi dice di essere. Ecco che qui entra in gioco un attacco **man in the middle**. Un terzo soggetto potrebbe intercettare le comunicazioni, fingendosi ad esempio come Alice. Esso riceve B calcolato da Bob e manda un suo C ad Alice. I due crederanno di star comunicando tra di loro, in realtà ci sono due comunicazioni separate verso C!



**RSA Key Exchange**

Questo protocollo è ibrido, nel senso che usa sia la crittografia simmetrica che quella asimmetrica. In particolare, viene usata la crittografia asimmetrica per fare l’encription, e poi viene trasportato in questo oggetto, in questo crittotesto, una chiave simmetrica kab. Supponiamo che A voglia condividere un segreto con B. A manda il segreto kab, codificato con la chiave pubblica del destinatario, kb, e grazie alla crittografia asimmetrica, questo segreto viaggerà in maniera confidenziale. Ricevuto questo crittotesto, B (e solo B può farlo), leggerà il segreto inventato da A. Il crittotesto è anche chiamato **busta digitale**.



**Certificazione**

Il problema della crittografia asimmetrica è (ogni volta che codifichiamo con una chiave pubblica) quello di sapere a chi appartenga quella chiave. Se sbagliamo in questo punto, c’è un attacco di segretezza sulla busta digitale. Questo problema si risolve con la certificazione. Tutti i certificati suggellano associazioni. Il certificato nel mondo digitale è un **certificato digitale**. Un certificato è **firmato**. Nel nostro caso, l’autorità si chiama autorità di certificazione. Il formato standard si chiama **X.509**.

Un certificato digitale è un’associazione tra il nome dell’agente e una stringa di bit. Questo certificato va poi firmato, quindi è garantita l’autenticazione e l’integrità. I certificati hanno una scadenza.



**Public-Key Infrastructure (PKI)**

Per risolvere i problemi di certificazione, non basta dire “ho il certificato”, ma bisogna dimostrarlo, usarlo. Ecco perché il protocollo deve usare il certificato quando richiesto, e questo protocollo si chiama **PKI**. Inoltre, ogni autorità di certificazione è certificata a sua volta da una precedente. Seguendo questo principio arriveremo a ciò che viene chiamata **root certification authority** (**RCA**).

Un’infrastruttura a chiave pubblica o sistema di certificazione consiste nella gerarchia di autorità e nelle tecnologie che esse usano.

**Chain of Trust**

La catena di fiducia di A è la lista completa dei certificati fino a quello di root, che permettono, insieme alla chiave pubblica della RCA, di verificare il certificato di A.

Il certificato **foglia** è così firmato:



L’autorità di certificazione è certificata da una prima di essa:



La RCA si **autofirma** da sola.

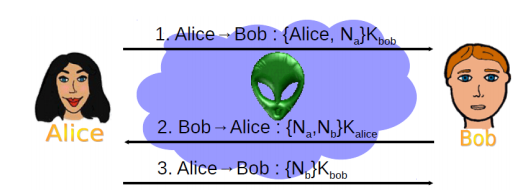


In caso un’autorità “i” viene corrotta, non è detto che tutto il sottoalbero sia corrotto. Infatti, se viene scoperta la chiave privata di un’autorità non è certo che sia scoperta la chiave privata dei suoi figli. In questo caso viene garantita in un certo modo la segretezza. Non è così invece per la fiducia (trust), la quale viene persa. Infatti, se un’autorità è corrotta avremmo tutti dei dubbi sulle autorità discendenti.

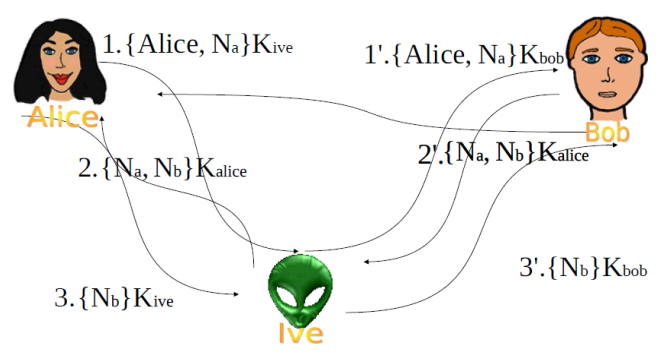
**Certificate Revocation List (CRL)**

Un CRL è un elenco di certificati digitali che sono stati revocati dalla Certification Authority prima della data di scadenza pianificata.

**Protocollo Needham-Schroder asimmetrico**

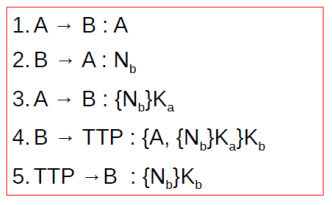


Un protocollo molto semplice, all’inizi della crittografia asimmetrica. Alice dichiara chi è e crea la nonce per Bob. Lui la apre, rispedisce la nonce indietro (freshness) e allo stesso tempo vuole anche lui garanzie e manda la sua nonce. Alice, quindi, manda la nonce di Bob indietro. Adesso i due sono autenticati. È andato bene finchè qualcuno non “scoprì” che è un protocollo vulnerabile ad un attacco **man-in-the-middle**. Prendendo un attaccante DY, anche se abbiamo crittografia, si riesce comunque a rompere il protocollo. Questo succede quando si hanno più sessioni intrecciate. Per esempio, se Alice sta parlando con Bob e con Ive, manderà la nonce a Ive, ma Ive si impersonificherà come Alice e manderà la stessa nonce a Bob facendo credere di essere Alice. Bob invierà ad Alice la sua di nonce. Alice rimanderà indietro la nonce di Bob come conferma, credendo però di star parlando con Ive e che quindi la nonce sia sua. Ive, a questo punto, è in possesso di entrambe le nonce.



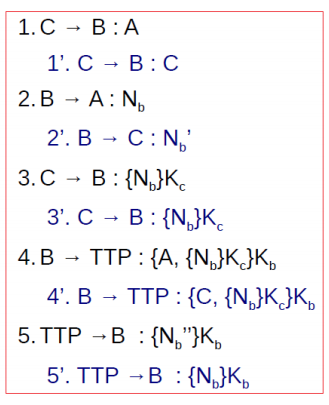
**Protocollo Woo-Lam**

Esso è un protocollo per crittografia simmetrica. Come sappiamo, in quella simmetrica, si ha una sola chiave. Ci vuole quindi un server di fiducia detto TTP (Trusted Third Party), che conosce tutte le chiavi.



Alice dice di essere Alice. Bob manda la sua nonce, mentre Alice la rimanda indietro criptata con la sua chiave. Bob chiede al server se è veramente Alice, chiedendo di provare a decrittare la risposta con la chiave di Alice. Il server riesce e manda la risposta indietro a Bob con successo.

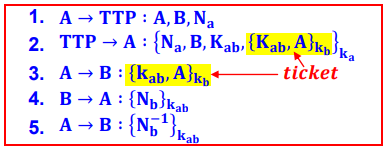
Questo protocollo è però vulnerabile fin dai primi passi.



L’attaccante apre due sessioni in contemporanea con Bob, una in cui dice di essere Alice e l’altra in cui dice di essere Charlie. A questo punto Bob manderà due nonce, una per ogni utente. Charlie butta la nonce per sé e cripta quella per Alice due volte, e la manda in entrambe le sessioni. Bob chiede al server di decrittare quella di Alice con la sua chiave e quella di Charlie con la sua. Il server rifiuta quella con la chiave di Alice (visto che è criptato con la chiave di Charlie) e manda indietro la risposta decrittata con la chiave di Charlie, che però conteneva la nonce per Alice! Bob tutto questo non lo sa, e quindi crede di star parlando con Alice.

Questo problema è facilmente risolvibile, basterebbe che il server dica a Bob **di chi è la chiave** che ha utilizzato per decrittare quella nonce.

**Symmetric Needham-Schroder**

****

Alice chiede una sessione con Bob a TTP. Esso crea una chiave di sessione che verrà mandata ad Alice, insieme alla nonce, ed inoltre le verrà inviato un “ticket”, ovvero un crittotesto da mandare a Bob per partecipare alla sessione. Il ticket è ovviamente criptato con la chiave a lungo termine di Bob; quindi, c’è garanzia che viene dal TTP. Dopo aver visto la chiave di sessione, Bob vuole essere sicuro che ci sia ancora qualcuno lì dietro che usa quella chiave; quindi, manda una nonce che gli verrà rispedita indietro.

**IPSec**

IPSec sta al livello IP e rende sicuro il livello IP. La sicurezza è un’applicazione. Immaginiamo di riuscire a codificare il traffico in uscita che va da noi verso un altro utente e che utilizziamo questo traffico codificato ad esempio facendolo passare verso altri utenti. Poiché il traffico è protetto, se lo facessimo passare verso nodi sicuri, nodi malevoli, non è un problema, poiché non riusciranno ad appropriarsi delle informazioni, ad utilizzarli. Questo è quello che fa, ad esempio, HTTPS. Quindi la sicurezza, in un certo senso, sta a un livello più alto.

IPSec distribuisce una **chiave di sessione** e questa chiave di sessione codifica i pacchetti. IPSec comprende una suite protocolli per garantire segretezza, autenticazione e integrità a livello IP:

* IKE (Internet Key Exchange): Distribuisce la chiave di sessione
* AH: Usa la chiave di sessione per fare l’autenticazione
* ESP: usa la chiave di sessione per ottenere la segretezza

Il concetto di **Security Association** (in breve **SA**) è alla base del funzionamento di IPsec. Una SA è un "contratto" fra le due entità coinvolte nella comunicazione; in essa vengono stabiliti i meccanismi di protezione e le chiavi da utilizzare durante il successivo trasferimento dei dati. Nell'ambito di IPsec, stabilire le security association è compito del protocollo IKE.

Una peculiarità delle SA è che individuano una comunicazione unidirezionale; quindi, durante la creazione della connessione le entità coinvolte creano e gestiscono una SA per ognuno dei versi della comunicazione; quindi, 2 SA individuano un canale full-duplex. Al fine di semplificare la gestione delle SA, viene utilizzato un apposito database detto **SAD** (**S**ecurity **A**ssociation **D**atabase), dove viene tenuta traccia delle SA attive. In particolare, una SA è costituita dai seguenti parametri:

* Gli indirizzi IP dei peer coinvolti nella comunicazione.
* Il protocollo che verrà utilizzato (AH o ESP)
* Un intero a 32 bit chiamato SPI, Security Parameter Index, che dirà come trattare il pacchetto tra i “nodi” in cui passerà.

Il security association database (SAD) è un database utilizzato dall’intero SPI per accedere alle security association interessate. Ogni riga contiene

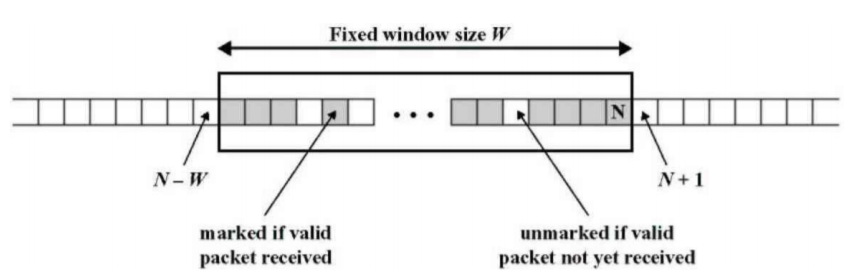
* AH info
* ESP info
* Durata SA

Dall'esame dei parametri di una SA si deducono quindi tutte le informazioni necessarie per stabilire le modalità in cui il traffico debba essere protetto; il passo successivo è definire *quale* traffico debba essere protetto: di questo si occupa la **Security Policy** (in breve **SP**). Una SP è una regola che stabilisce che tipo di traffico deve essere instradato nel tunnel e quindi essere coperto da IPsec; in modo analogo alle SA, le SP sono contenute in un SPD (**S**ecurity **P**olicy **D**atabase).

**Authentication Header** (abbreviato **AH**), è un protocollo che fa parte della suite IPsec. Il suo compito è quello di fornire un controllo di integrità pacchetto per pacchetto, verifica dell'autenticità del mittente e protezione contro i replay attack. AH non garantisce in alcun modo la segretezza del messaggio. L'autenticità è garantita tramite funzioni di hash a chiave simmetrica, ossia tramite il meccanismo delle pre-shared keys. Per poter comunicare, due entità devono condividere la medesima chiave; tale chiave viene combinata con il messaggio originale e quindi viene calcolato il checksum tramite una funzione di hash crittografico (MD5 o SHA). Il messaggio e il checksum vengono, infine, inviati al *peer* remoto. Il *peer* remoto riceve il messaggio; dato che questo è in chiaro, lo può leggere, combinare con la chiave di cui è a conoscenza e calcolare il checksum. Se il checksum corrisponde a quello inviato, il messaggio è autentico e viene accettato altrimenti viene scartato in quanto è stato modificato in un modo non consentito dallo standard.Il protocollo AH è progettato per proteggere l'intero pacchetto IP inviato; tuttavia bisogna considerare che alcuni campi dell'header IP, come il **TTL**, variano durante la trasmissione; queste modifiche devono essere necessariamente consentite, per cui prima di calcolare il checksum, i campi cui è permesso variare vengono posti a **0**.

**Meccanismo a finestra**

Per proteggersi contro i replay attack si utilizza il Sequence Number di AH. In particolare, se esso cade entro un certo range di numeri allora verrà accettato. Per questo, utilizziamo il “modello a finestra”.

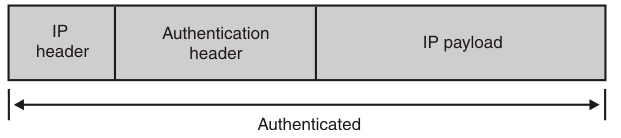


Se il numero X cade prima di N-W allora il pacchetto è troppo vecchio e viene scartato. Se il numero cade dentro la finestra, verrà accettato e segnato (così se ne arriva un altro uguale viene scartato visto che è stato già accettato). Se il numero è più alto della finestra, la finestra verrà shiftata in avanti.

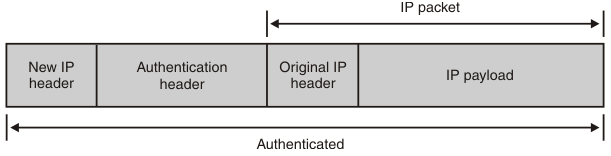
**AH Trasporto vs Tunnel**

A seconda della modalità di funzionamento di IPsec (tunnel mode o transport mode), il pacchetto originale viene alterato in modo diverso.

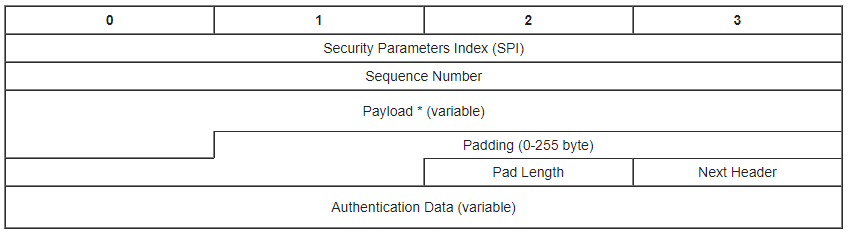
In modalità trasporto, il pacchetto mantiene gli IP originali, quindi si vedono in maniera chiara la sorgente e la destinazione del pacchetto.

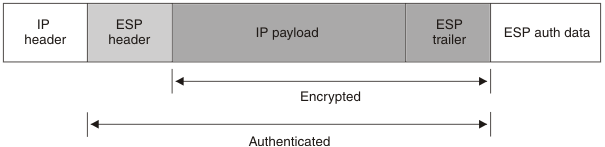


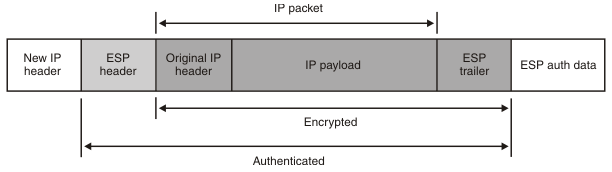
In modalità tunnel dei nuovi IP vengono usati esternamente, ovvero gli endpoint sicuri. In questo modo non solo viene criptato il payload, ma anche gli IP veri e propri.



**ESP Trasporto vs Tunnel**

In aggiunta ad AH, questo protocollo fornisce segretezza.



****

Una differenza importante sta nel fatto che AH è incompatibile con NAT, visto che esegue il controllo di integrità anche sugli IP, quindi non possono essere modificati. Cosa che, invece, ESP non fa.

**IKE**

È il protocollo che si occupa della distribuzione della chiave, ed è la “somma” di

Oakley, ovvero una variante di Diffie-Hellman potenziata che fornisce la chiave e di ISAKMP (Key management protocol) che espande il segreto iniziale in un segreto più lungo in termini di bit.

**Intrusion Detection System (IDS)**

Spesso siamo incappati nella differenza, empiricamente, tra virus e intruso. Con intruso si intende un processo alieno; si intende anche un processo che riesce ad acquistare determinati privilegi.

**Intrusione versus Malware**

La caratterizzazione tipica di Malware è di infezione su file. La caratterizzazione tipica di intrusione è di processo in esecuzione.

**Negazione del servizio (DoS)**

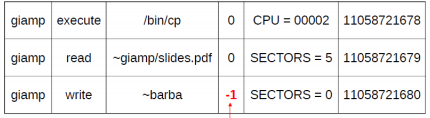
Attacco che impedisce la normale operatività di un sistema, degradandola o bloccandola del tutto. Definiamo tre tipi:

* cDoS: consumo delle risorse computazionali
* mDoS: consumo della memoria
* bDoS: consumo della banda di trasmissione

**Intrusione**

Ottenere illecitamente privilegi superiori a quelli posseduti lecitamente.

Per vedere se ci sono delle intrusioni possiamo fare dell’auditing, cioè, log di sistema. In un log possiamo vedere se ci sono state operazioni illecite, per esempio, da parte di chi e cosa ha fatto. Un modello di record, ovvero di una riga di log, è quello di Denning.

Esso contiene: Soggetto, Azione, Oggetto, Eccezione, Risorse, Timestamp.

**Tecniche di rilevamento**

Tecniche assolute (indipendenti dal passato): massimo 3 fallimenti di inserimento di password.

Tecniche dipendenti dal passato: si osserva il funzionamento e si assume che il futuro sia come il passato. Questo somiglia molto alla memoria cache.

Il rilevamento può essere statistico o a regole. Il rilevamento statistico crea un modello di utente “buono”: se ci distacchiamo da questo modello viene segnalato un attacco. Il modello a regole ha ben specifiche regole.

**Firewall**

Dispositivo che controlla il flusso di traffico fra reti con diverse impostazioni di sicurezza. Il firewall, per buona norma, dovrebbe trovarsi su una macchina addennizzata, ovvero un **bastion host**, adeguato solamente per contenere il firewall.

Un packet filtering router fa filtraggio di pacchetti. Esso lavora a livello di rete, quindi vede gli IP. Esso può stabilire delle regole di blocco o di accesso da determinati IP ad altri IP. La debolezza del firewall è che con un ip spoofer è possibile bypassarlo. Ecco perché se gli si accoppia IPSec esso può essere molto efficace.

**SSL**

(vedere prima fogli Vito)

Una sessione contiene diverse informazioni, tra cui: Session ID, Certificato, Metodo di compressione, Cipher spec, Master secret, is resumable flag. Attenzione, se si ripristina una sessione, si salta la fase 2 e 3 perché hanno già il master secret! Si eseguirà solo la fase 1 e la fase 4 per calcolare le chiavi per la cifratura e MAC con le nuove nonce.

Il carico maggiore viene eseguito dal **Record Protocol**. Esso prende i dati che l’applicazione vuole trasmettere, li frammenta, li comprime e aggiunge il MAC (il MAC dà autenticazione e integrità). Il tutto viene cifrato e viene aggiunto un Record Header.

Immagine che contiene testo, lavagnabianca

Descrizione generata automaticamente

Il MAC ha una “formula” ben precisa ed è la seguente:

Immagine che contiene testo

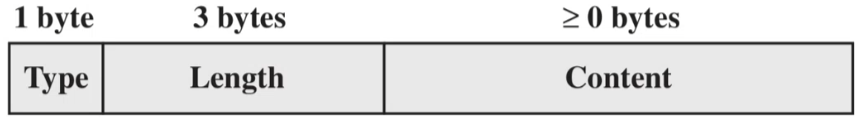
Descrizione generata automaticamente

(L’hash può essere MD5 o SHA-1. Ovviamente questo perché è un protocollo datato, quindi ai tempi c’erano questi disponibili).

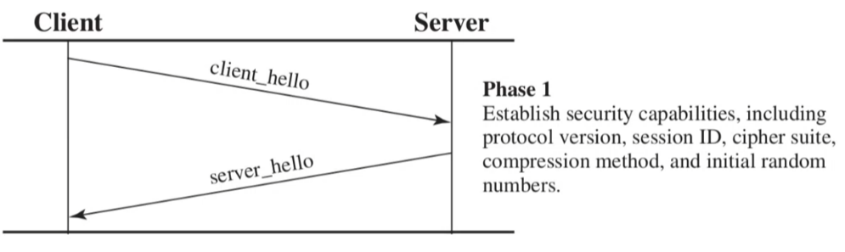
Il **Cipher Suite** è un campo che determina che tipo di crittografia utilizzare.

L’ **Alert Protocol** dice solamente se ci sono errori. Gli errori possono essere di due tipi: warning e fatal. Se un errore è fatale, la connessione viene interrotta e non può essere ripristinata (IsResumable set to 0).

L’ Handshake Protocol è quello che sancisce la modalità di comunicazione, le chiavi ecc… Un suo messaggio è tipicamente formato così:



Esso si divide in fasi:



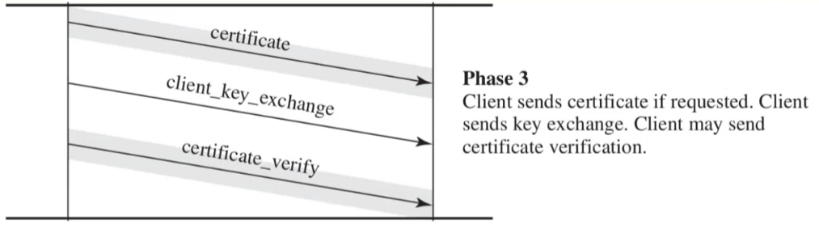
Il client manda il suo messaggio, specificando la versione del protocollo a cui può sottostare, i metodi di crittografia che può usare, una nonce random, un session ID, il cui valore sarà 0 se vuole una nuova sessione, diversa da 0 se ne vuole ripristinare una chiusa precedentemente. Essendo un accordo “a due vie”, il server prova ad adattarsi alle esigenze del client, ed invia le stesse cose, anche la sua nuova nonce.

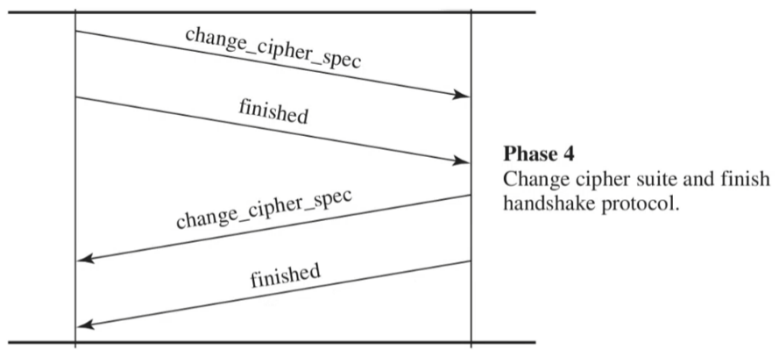
Importante: la **cipher suite è l’insieme delle tecniche** (crittografia + hash + scambio di chiave), mentre la cipher spec non è altro che l’unione dei primi due elementi.

Immagine che contiene tavolo

Descrizione generata automaticamente

La parte in grigio è opzionale, ma renderla opzionale sminuisce il protocollo.





Change\_cipher\_spec significa che per il client e il server si sono accordati sulla cipher spec. Il “finished” è un hash value.

Quindi, la fase 1 serve a mettere le due parti d’accordo sui parametri fondamentali. Le fasi centrali utilizzano uno scambio di certificati per realizzare lo scambio della chiave. La fase 4 conferma il lavoro fatto con la cipher spec scelta.

Master Secret

Immagine che contiene testo

Descrizione generata automaticamente

Esso fa uso delle nonce del client e del server e dei caratteri per creare “disordine” (entropia) nell’hash.

Immagine che contiene testo

Descrizione generata automaticamente

Il key\_block è un pezzo di chiave che si differenzia dalla generazione del master secret dal fatto che invece del pre-master secret qui viene utilizzato il master secret. E cosa più importante, il blocco non si ferma a 48 byte come il master secret, ma ci sono dei puntini, ovvero continua finché si ritiene necessario aggiungere byte.

**TSL**

TLS nasce nel 1999 ma diventa standard con la sua versione 3.0. A causa della vecchia data di creazione, le funzioni hash che vengono utilizzate sono ancora MD5 e SHA1. Invece del MAC utilizziamo l’ HMAC che è più veloce ma ha lo stesso scopo del precedente.

HMAC:



K è la chiave, M è il messaggio, K+ è la chiave con l’aggiunta di zeri a sinistra, tale che la lunghezza dei bit è uguale a quella dell’hash (512). Il pallino si chiama “exclusive or”.

Pseudo random function (PRF):

E’ la funzione che ci permette di costruire il materiale crittografico “casuale”. Essa prende in ingresso un segreto (pre o master secret), una label (“master secret” se dobbiamo calcolare l’ MS o “key block” per calcolare il key block) e un seme.

Immagine che contiene testo

Descrizione generata automaticamente

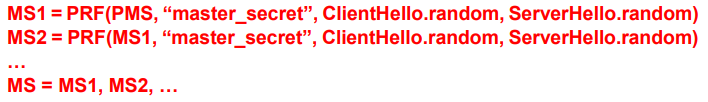
Quella P è una funzione P\_Hash che prende due parametri, un segreto ed un seme. Essa è una funzione ricorsiva, infatti una volta calcolata la prima P\_Hash, possiamo usarla come seed per la seconda e così via. Il caso base è il seed che decidiamo noi.

Immagine che contiene testo

Descrizione generata automaticamente

Novità TLS 1.2 (2008): E’ possibile scegliere quale funzione hash usare (non più solo MD5 e SHA che sono risultate deboli).

Master Secret in TLS:



Così via fino a raggiungere 48 byte.

Key block in TLS:

Immagine che contiene testo

Descrizione generata automaticamente