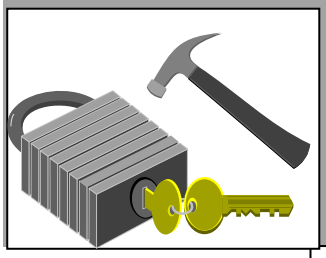
Capitolo 6



Servizi sicuri

Per conseguire un particolare obiettivo di sicurezza occorre in generale ricorrere ad un **servizio sicuro**, cioè all'uso coordinato di più meccanismi per la sicurezza.

Molti sono i servizi disponibili e facili da usare: in certi casi sono già presenti nel Sw del calcolatore, in altri casi occorre prima installarli. Nel seguito esamineremo alcune realizzazioni di notevole interesse, distinguendo due categorie: quella dei servizi che è possibile chiamare in causa dal livello applicazione (Kerberos, PGP, TSS) e quella dei servizi inseriti o inseribili all'interno della suite di protocolli di rete (SSL/TLS, IPv6, WEP).

Esistono tuttavia situazioni in cui l'uso del servizio presuppone lo svolgimento di attività complesse e situazioni in cui il servizio deve essere sviluppato ex-novo. In entrambi i casi, per fortuna, si dispone oggi di supporti efficaci e semplici da usare. Al termine di questo capitolo ne esamineremo due: il toolkit di OpenSSL e l'insieme di strumenti che il linguaggio Java mette a disposizione di chi deve sviluppare applicazioni sicure.

6.1 Servizi sicuri del livello applicazione

6.1.1 Kerberos

Kerberos è un servizio di autenticazione per un ambiente client/server[63]. Compito del sistema è il controllo d'accesso ad un insieme di server da parte di una comunità di utenti avente a disposizione un certo numero di workstation.

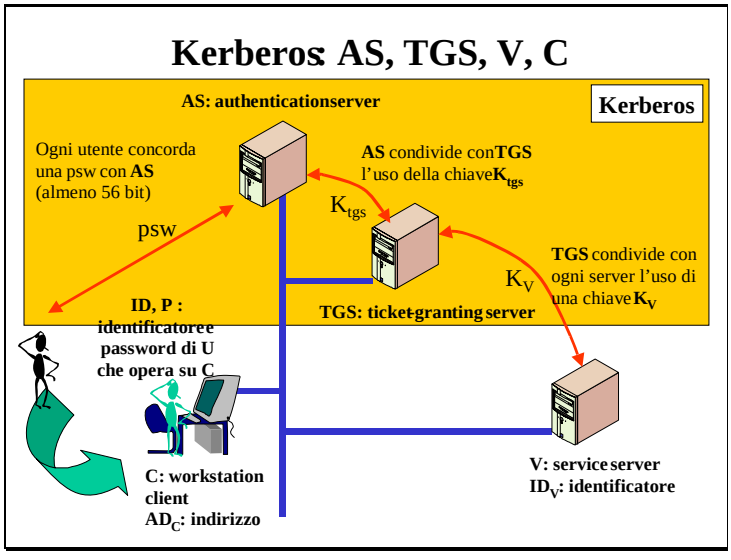
Il simbolo del cane a tre teste discende dall'idea originaria di realizzare un sistema centralizzato per *authentication*, *accounting* e *audit*. Il servizio di autenticazione, l'unico ad essere stato poi realizzato, si ispira al modello del KDC (v. pag. 76), impiega solo meccanismi a chiavi simmetriche e richiede la presenza in linea di una terza parte fidata.

Le prime tre versioni del servizio sono state sviluppate al MIT alla fine degli anni '80; l'attuale versione 5 costituisce uno standard Internet (RFC 1510).

Ogni utente del servizio ha un suo identificativo **ID** ed un'**unica password P**, che gli consente di accedere a tutti i servizi. Il sistema che controlla gli accessi memorizza un'impronta **H(P)** di 64 bit come termine di paragone.

Gli obiettivi di sicurezza sono:

- rendere impossibile ad un intruso l'impersonare un utente legittimo;
- eliminare a priori la comunicazione della password;
- rendere trasparente all'utente l'intero processo di autenticazione;
- richiedere che l'utente fornisca prova della sua identità e del suo diritto al primo accesso ad ogni servizio;
- consentire agli utenti una sicura identificazione dei server.



Efficienza, affidabilità e scalabilità del servizio di autenticazione sono ottenute tramite due server particolari: **AS** e **TGS**.

AS (*Authentication Server*): memorizza la password ed i diritti di tutti gli utenti, lancia una sfida d'identificazione a chi inizia una sessione di lavoro presso una qualsiasi delle workstation e gli restituisce un documento cifrato che dovrà presentare a **TGS** al fine di ottenere specifiche autorizzazioni.

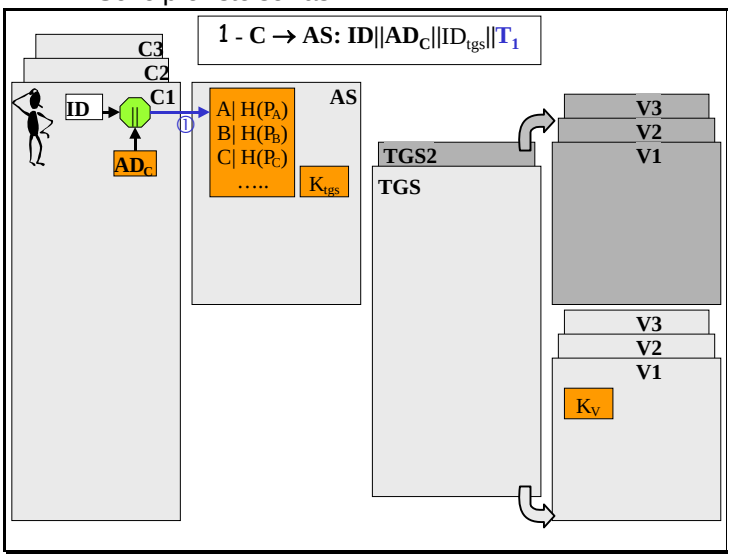
TGS (*Ticket-Granting Server*): condivide una chiave segreta con **AS** ed una con ogni altro server, decodifica e verifica il documento di **AS** che l'utente gli invia, esamina la correttezza della risposta alla sfida lanciata da **AS** e, se tutto è "a posto", restituisce all'utente un diritto d'accesso (*ticket*) al server **V** di suo interesse, valido per tutta la sessione di lavoro.

[63] Una trattazione più dettagliata è presente in [6], cap.14

Il servizio vero e proprio inizia solo dopo che l'utente ed il server si sono reciprocamente identificati.

1 - Avvio di una sessione di lavoro

Sono previste sei fasi.



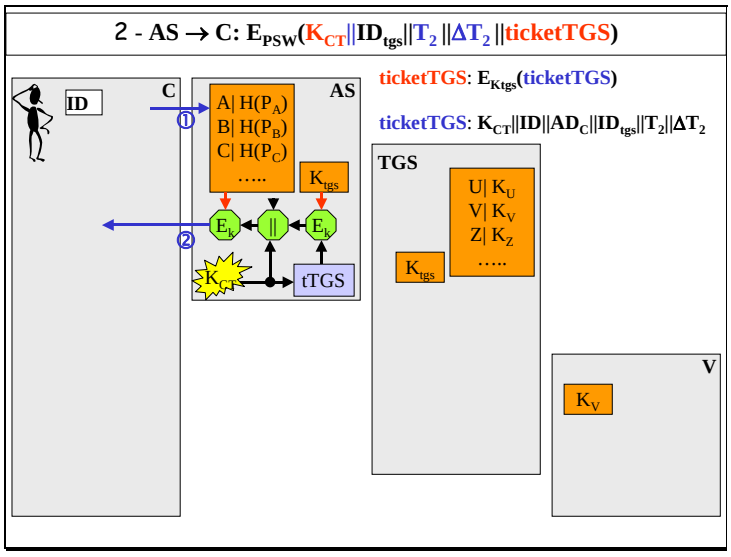
1 - Quando un utente inizia a lavorare su una workstation (nel seguito indicata con **C**), le deve prima di tutto fornire il suo identificativo e quello del server **TGS** da cui vuole ottenere un ticket (i server, infatti, possono essere stati suddivisi in diversi gruppi, o ***realm***, ciascuno controllato da un suo **TGS**).

Una volta conosciuti questi dati, la stazione **C** provvede ad inviare ad **AS** la richiesta d'accesso a **TGS**:

$$C \rightarrow AS: ID||ADc||ID_{tgs}||T_1$$

Simboli:

- **ID**: identificativo dell'utente operante su **C**,
- **ADc**: indirizzo di rete di **C**,
- ID_{tgs}: identificativo del server TGS,
- T_1: marca temporale di log-on.

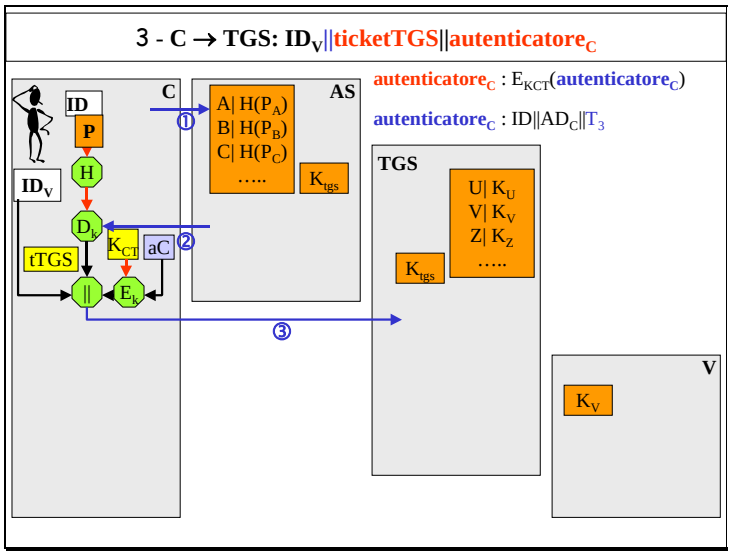


2 - **AS** controlla T_1, impiega **ID** per prelevare dalla sua memoria **H(P)** (nel seguito indicata da **PSW**) e lo sfida a mettere in chiaro un messaggio cifrato con questa chiave.

$$AS \rightarrow C: E_{PSW}(K_{CT}||ID_{tgs}||T_2||DT_2||\ ||E_{KTGS}(K_{CT}||ID||ADc||T_2||DT_2))$$

Simboli:

- K_{CT}: chiave di sessione per le successive comunicazioni tra **C** e **TGS**,
- T_2, DT_2: inizio e durata massima della sessione di lavoro per **ID** su **C**,
- K_{TGS}: chiave segreta che **AS** condivide con **TGS** e che impiega per cifrare un *ticket* da cui **TGS** potrà dedurre chi è l'utente, su quale stazione lavora, qual è la chiave di sessione e per quanto è valida.



3 - **C** richiede all'utente di digitare la sua password **P**, ne calcola l'hash e lo impiega come chiave per decifrare il messaggio di **AS**.

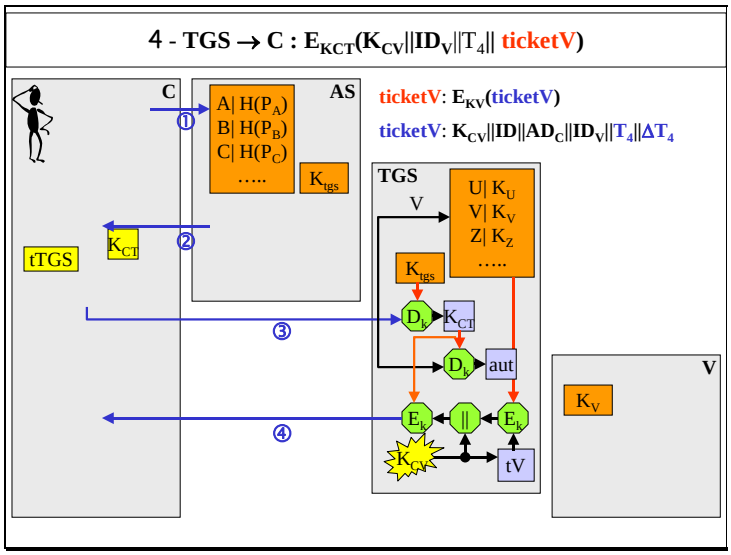
Se la prova ha esito positivo, **C** chiede all'utente l'identificativo **IDv** del server **V** cui vuole accedere e lo invia a **TGS** unitamente

- al ticket cifrato con K_{tgs} che **AS** ha predisposto per **TGS**,
- ad un "autenticatore" cifrato con K_{CT} con cui l'utente dimostra di essere riuscito a superare la sfida lanciata da **AS**.

$$C \rightarrow TGS: IDv||\ ||E_{KTGS}(K_{CT}||ID||ADc||T_2||DT_2)||\ ||E_{KCT}(ID||ADc||T_3)$$

Simboli:

- T_3: timestamp che consentirà a TGS di controllare se la sessione è ancora valida.

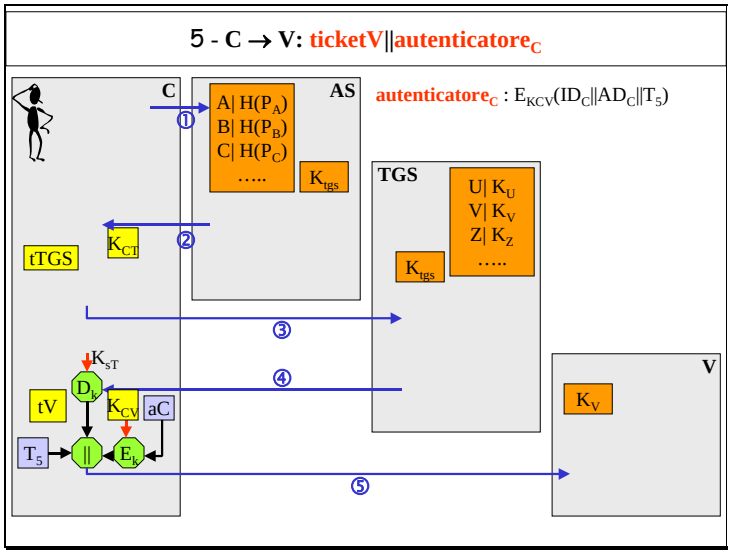


4 – **TGS** decifra il ticket con chiave K_{TGS}, prende atto delle indicazioni fornite da **AS** ed estrae la chiave di sessione K_{CT}.

Decifrando con questa chiave l'autenticatore allegato al messaggio che ha ricevuto da **C**, **TGS** completa il protocollo a sfida/risposta lanciato da **AS**: se l'autenticatore contiene le informazioni del ticket, l'utente è, infatti, proprio chi dice di essere.

TGS sceglie allora a caso una nuova chiave di sessione K_{CV} per il colloquio tra **C** e **V**, fissa un intervallo di tempo entro cui autorizza l'accesso a **V** e predispone un ticket per **V** con la chiave segreta K_V che condividono; l'intero messaggio è inviato a **C** cifrato con K_{CT}.

$$\mathbf{TGS \rightarrow C: E_{KCT}(K_{CV}||ID_V||T_4|| ||E_{KV}(K_{CV}||ID||ADc||T_4||DT_4))}$$

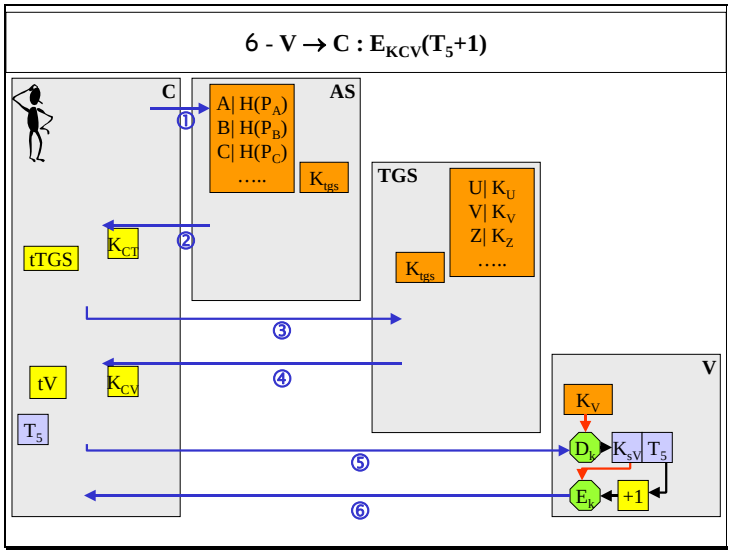


5 - **C** decifra il messaggio di **TGS** con la chiave di sessione K_{CT} che ha memorizzato al passo 2. A questo punto **C** viene dunque in possesso del ticket d'accesso a **V** e conosce anche la chiave di sessione K_{CV} da usare per comunicare con lui.

C inoltra a **V** il ticket cifrato predisposto da **TGS** ed un autenticatore con cui da un lato si fa identificare, da un latro lato sfida il server a dimostrare che è proprio **V**.

$$\mathbf{C \rightarrow V: E_{KV}(K_{CV}||ID||ADc||T_4||DT_4)|| ||E_{KCV}(ID||ADc||T_5)}$$

La marca temporale T_5 indica il momento in cui **C** inizia a contattare **V**. Il diritto d'accesso scadrà all'istante $T_5 + DT_4$



6 - **V**, se è veramente lui, è in grado di decifrare il ticket di **TGS** inoltratogli da **C**. Può in particolare estrarre la chiave di sessione K_{CV} ed impiegarla per mettere in chiaro l'autenticatore allegato al ticket.

Se tutto va bene **V** è certo dell'identità dell'utente.

Per farsi a sua volta identificare **V** genera un messaggio con cui dimostra a **C** di aver decifrato quanto gli ha appena inviato.

$$\mathbf{V \rightarrow C: E_{KCV}(T_5+1)}$$

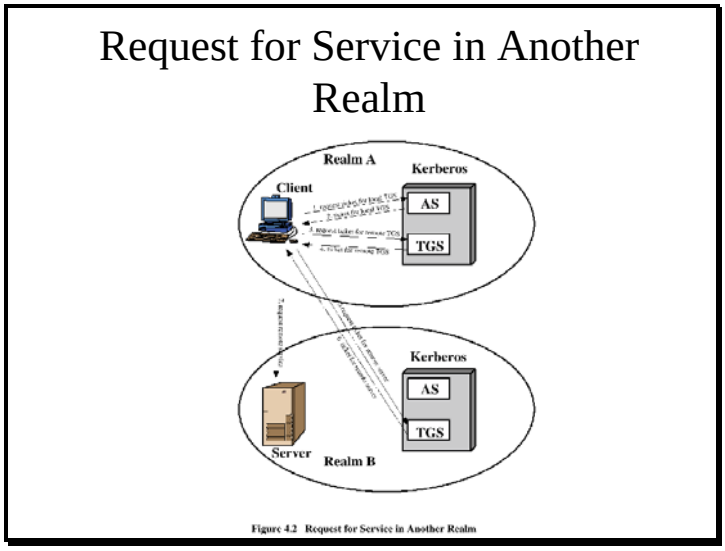
C decifra, controlla e, se tutto va bene, si convince di essere connesso proprio con il server **V**.

A questo punto il **client C** inizia ad avvalersi dei servizi del **server V**.

Se successivamente l'utente ha ancora bisogno di **V** il protocollo ricomincia dal **passo 5**.

Se durante la sessione l'utente ha bisogno di accedere anche ad un altro server, il protocollo deve essere riavviato dal **passo 3**.

2 - Servizio inter-realm



Per rendere scalabile questo servizio di autenticazione, e' stata prevista la coesistenza di diversi Kerberos tra cui esiste un rapporto di reciproca fiducia.

Se un utente di un realm ha bisogno di accedere ad un servizio inserito in un altro realm e gestito quindi da un altro TGS a lui noto, ne indica l'identificativo al posto di idv, quando, al **passo 3**, si mette in contatto con il suo TGS:

C → TGS: ID_{TGSREM}||ticket||autenticatore

Simboli:

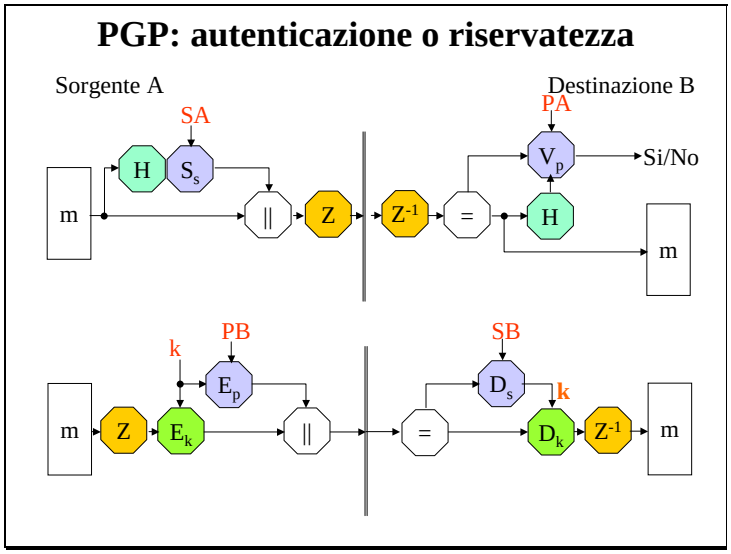
- **ID_{TGSREM}**: identificativo del server TGS dell'altro realm.

Al **passo 4** C ottiene il ticket d'accesso al TGS remoto, al **passo 5** gli richiede il ticket d'accesso al server desiderato, al **passo 6** lo ottiene ed al **passo 7** contatta il server.

6.1.2 PGP[64]

Il **PGP** (*Pretty Good Privacy*), inventato da Philip R. Zimmerman nel 1993, è un servizio per la riservatezza e l'autenticazione, impiegato in applicazioni di posta elettronica e di archiviazione di file. Scritto in C ed in C++, è free software (ad esclusione della versione più recente, che può essere solo provata), opera su un'ampia gamma di piattaforme, usa algoritmi con sicurezza ben nota e non dipende da organizzazioni governative o similari.

1 - Autenticazione e Riservatezza



L'**autenticazione** è ottenuta allegando al documento in chiaro un suo hash firmato; gli utenti possono scegliere di farlo o con RSA, o con DSS. Il **PGP** consente anche di "zippare" il documento autenticato, per renderne efficienti la memorizzazione e la trasmissione.

Per garantire la **riservatezza**, il PGP impiega il principio del Cifrario ibrido (v. pag. 107). Il mittente usa un Cifrario simmetrico (TDES, IDEA, CAST-128, AES) con modalità CFB per trasformare il documento; la chiave di sessione è comunicata o con RSA, o con ElGamal, a seconda del tipo di chiave pubblica (RSA o DH nella terminologia PGP) che il destinatario si è generato. Prima della cifratura è possibile comprimere il messaggio.

E' infine possibile proteggere contemporaneamente l'autenticità e la riservatezza: in questo caso il documento è prima firmato, poi cifrato. Chi lo riceve deve dunque prima decifrare e poi verificare.

2 - Gestione delle chiavi

Il PGP si prende totalmente in carico la gestione delle chiavi dell'utente e dei suoi corrispondenti: a tal fine impiega un **generatore di numeri casuali**, un **generatore di numeri pseudocasuali** e **due portachiavi**.

La generazione di numeri casuali si basa su una continua raccolta di eventi verificatisi nell'applicazione (pause dell'utente tra una digitazione e l'altra, caratteri della passphrase, ecc.); i dati così ottenuti vengono dapprima sottoposti ad operazioni di EX-OR ed al calcolo di impronte e poi alloggiati in un archivio, detto ***Random-Pool***. Da tale archivio, il PGP estrae la **chiave privata** dell'utente ed il **seed** per avviare in maniera imprevedibile il generatore di numeri pseudocasuali.

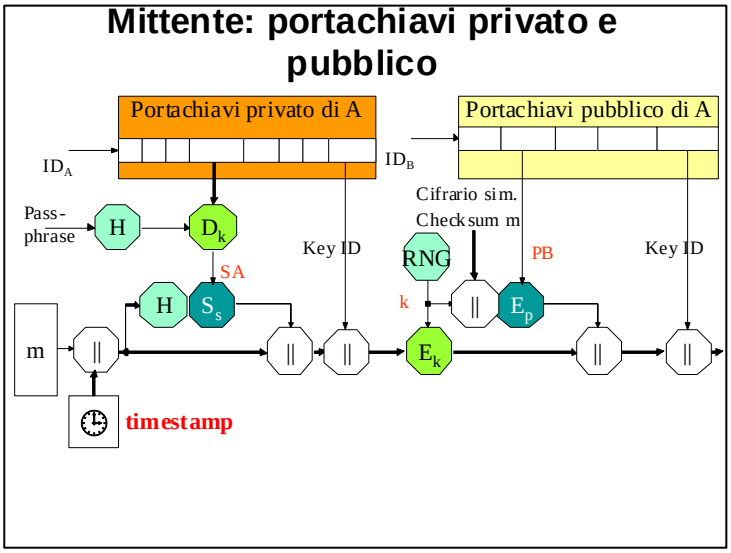
Il generatore pseudocasuale del PGP si basa sullo standard ANSI X9.17 (v. pag. 31) e viene chiamato in causa ogniqualvolta occorre una **chiave di sessione** o un **vettore d'inizializzazione**.

Ogni utente viene infine dotato di **due distinti portachiavi**. Nel **portachiavi pubblico** sono alloggiate le sue chiavi pubbliche e quelle dei suoi corrispondenti. Nel **portachiavi privato** sono alloggiate, **cifrate**, le sue chiavi

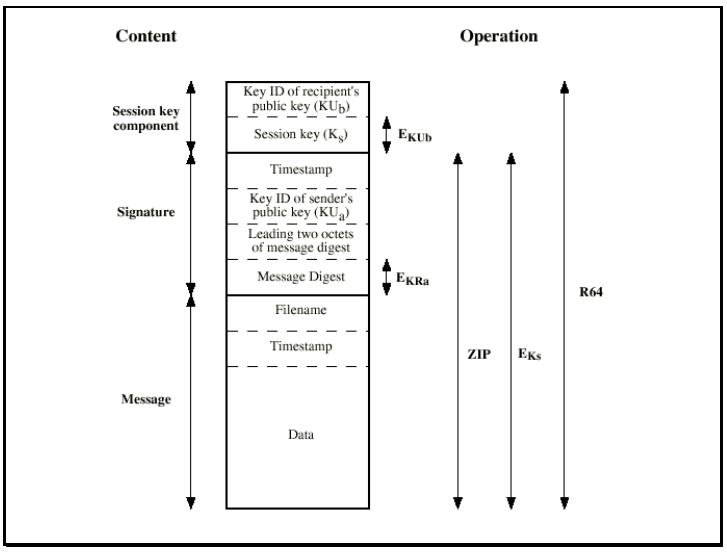
[64] Una trattazione più dettagliata è presente in [6], cap 15

private; la cifratura è fatta con un cifrario simmetrico (CAST-128) utilizzando come chiave 128 bit del hash della passphrase.

In figura è evidenziato l'uso dei portachiavi di un utente A, durante la firma e la cifratura di un messaggio: procedendo da sinistra verso destra si individuano sei attività consecutive.



1. Tramite il suo ID, viene estratta una chiave di firma dal portachiavi privato; per decifrarla viene usato il meccanismo simmetrico **D_k** e la chiave **H(passphrase)**.
2. Si concatena a **m** l'istante **t** di firma (*timestamp*)**,** se ne calcola l'hash e lo si firma con l'algoritmo asimmetrico **S_s**; il risultato è infine concatenato con **m||t** e con l'identificativo di chiave.
3. Tramite il suo ID, viene estratta dal portachiavi pubblico una delle chiavi di cifratura del destinatario.
4. Con la chiave pubblica e con il meccanismo asimmetrico **E_p** viene cifrata la chiave di messaggio **k**, unitamente al **nome** del cifrario sim. scelto ed alla **checksum** di m.
5. Con il cifrario sim. scelto si calcola **E_k(m')**.
6. I due crittogrammi sono concatenati tra loro e con l'ID della chiave del destinatario.



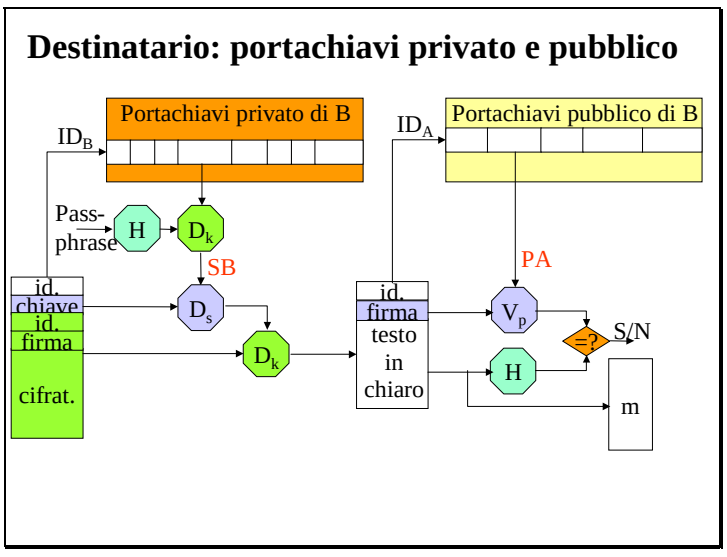
In figura è mostrato il formato di un messaggio PGP. La parte più in basso, sempre presente, contiene il nome del documento, la data di creazione ed il testo, o in chiaro o cifrato secondo la scelta operata dal mittente.

La parte in alto è presente se il messaggio è stato cifrato e contiene:

- l'identificatore della chiave pubblica del destinatario,
- il valore cifrato della chiave di sessione.

La parte sottostante è presente se il messaggio è stato firmato e contiene:

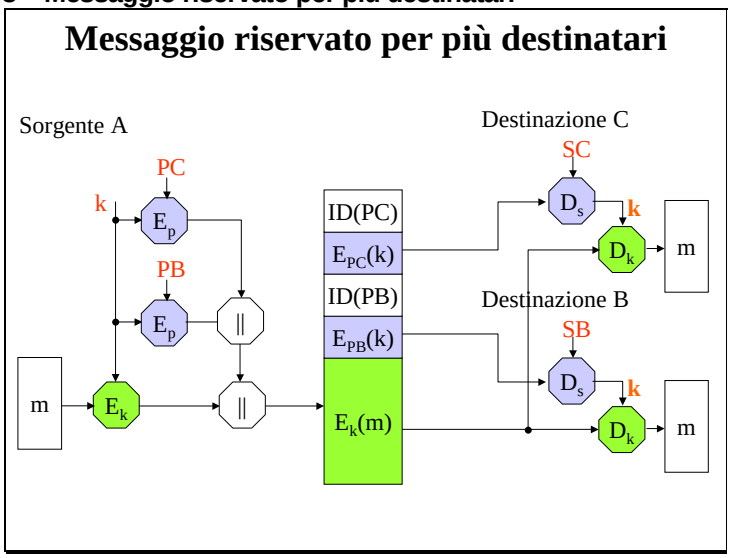
- l'indicazione del giorno/data/ora/ecc. in cui è stata eseguita la firma,
- l'ID della chiave pubblica del mittente,
- l'impronta firmata della parte "Message" preceduta dai suoi primi due byte in chiaro per agevolare i controlli del verificatore.



In ricezione, il messaggio è prima decifrato (a sinistra in figura) e poi verificato (a destra, con l'ipotesi di una firma RSA). Il PGP di B deve svolgere quattro attività.

1. Tramite l'identificativo di chiave in testa al messaggio, verifica se la chiave è presente nel portachiavi privato; in caso affermativo, richiede la passphrase e mette in chiaro SB.
2. Con il meccanismo asimmetrico **D_s** ottiene la session key e con il meccanismo simmetrico **D_k** mette in chiaro il testo.
3. Verifica se il portachiavi pubblico contiene la chiave pubblica di A: se PA esiste, la estrae e l'impiegata per mettere in chiaro, con **V_p**, l'hash del documento.
4. Calcola l'impronta del messaggio ricevuto e confronta il risultato con quello ottenuto al passo 3.

3 – Messaggio riservato per più destinatari



Una volta cifrato un messaggio con una certa chiave **k**, è possibile inviarlo anche a più corrispondenti.

A tal fine al testo cifrato occorre aggiungere tante intestazioni quanti sono i destinatari, contenenti ciascuna la cifratura asimmetrica della chiave one-time con l'appropriata chiave pubblica.

All'arrivo del messaggio, ogni destinatario, dopo aver identificato la parte dell'intestazione che lo riguarda, può al solito rimettere in chiaro dapprima **k** e poi **m**.

Un'applicazione interessante di questo modo di operare si ha quando il PGP è usato per l'archiviazione sicura di documenti.

Il proprietario del file system può, infatti, cifrare la chiave di cifratura di ogni file con due chiavi pubbliche, la sua e quella di un altro utente di sua fiducia, per garantire il recovery in sua assenza o nel caso di crash del suo portachiavi.

ESEMPIO – I progettisti che operano in un'Impresa salvano il loro lavoro quotidiano cifrandolo con la loro chiave e con quella del loro Responsabile Aziendale. In caso di dimissioni, l'Impresa può recuperare il lavoro fatto.

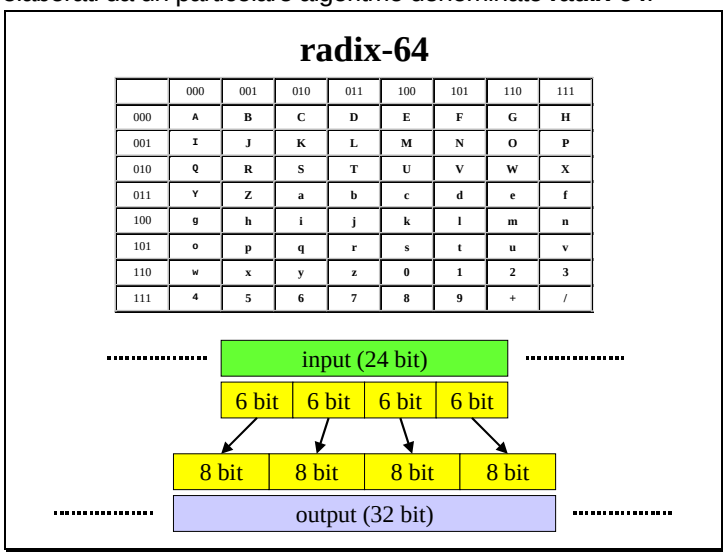
4 - Compatibilità con i servizi di rete

Il **PGP** è progettato per essere compatibile con i sottostanti servizi di rete. Interessano due aspetti:

- "come è fatto" il messaggio che deve essere trasmesso,
- "quanto è lungo".

Per quanto riguarda il "come è fatto" è necessario che i byte che compongono il messaggio, interpretati secondo il codice ASCII, rappresentino caratteri stampabili **e non** caratteri di controllo (questo vincolo è ad esempio imposto dallo strato **SMTP**, che i servizi di posta elettronica su Internet prevedono di avere subito sotto).

I file binari prodotti dai servizi per l'autenticazione e la riservatezza devono dunque essere ulteriormente elaborati da un particolare algoritmo denominato **radix-64**.



	000	001	010	011	100	101	110	111
000	A	B	C	D	E	F	G	H
001	I	J	K	L	M	N	O	P
010	Q	R	S	T	U	V	W	X
011	Y	Z	a	b	c	d	e	f
100	g	h	i	j	k	l	m	n
101	o	p	q	r	s	t	u	v
110	w	x	y	z	0	1	2	3
111	4	5	6	7	8	9	+	/

L'algoritmo radix-64

1. scompone il messaggio in tanti blocchi di 24 bit (3 byte) ciascuno,
2. scompone ogni blocco di 3 byte in 4 stringhe, ciascuna di 6 bit,
3. accedendo ad una tabella come quella indicata in figura, sostituisce ad ogni stringa di 6 bit il byte che, in ASCII a 8 bit, rappresenta il carattere stampabile cui deve corrispondere.

Ogni blocco diventa così di $4 \times 8 = 32$ bit; se necessario, l'ultimo blocco è completato con il simbolo di riempimento "=".

Complessivamente quindi la lunghezza del messaggio è incrementata del 33%.

ESEMPIO – Siano **000000–111111-110111-101011** i 24 bit in ingresso a radix-64.

- Dalla tabella della precedente figura si ottiene: **A / 3 r**
- La codifica ASCII determina i seguenti 32 bit di uscita: **01000001–00101111–00110011-01110010**

A valle di questo servizio di trascodifica il **PGP** si preoccupa della lunghezza del file risultante, provvedendo alla frammentazione dei messaggi troppo lunghi (ad es. > 50 KB) in trasmissione ed al riassemblaggio dei frammenti in ricezione.

6.1.3 TSS (*Time Stamp Service*)[65]

Diversi sono i programmi (del software di base e del software applicativo) che durante la loro esecuzione chiamano in causa l'orologio interno del calcolatore al fine di associare una **marca temporale** (anno, mese, giorno, ora, ecc.) ai dati che devono gestire.

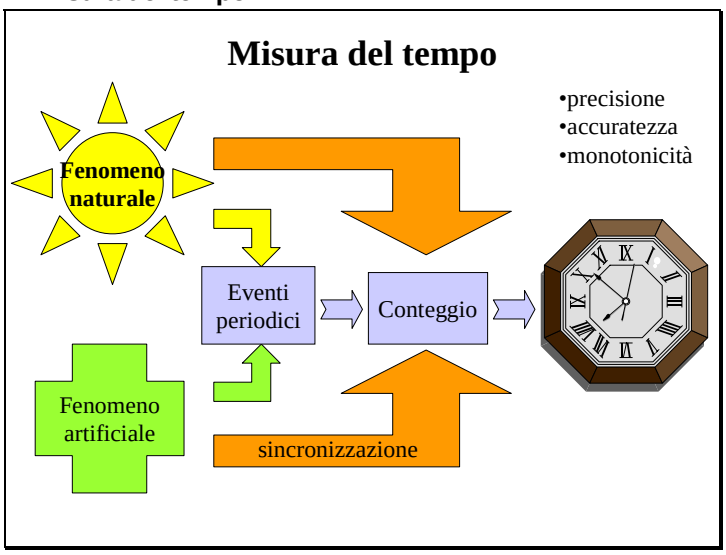
Diversa è anche l'accuratezza richiesta alle marche temporali: tutto dipende dall'uso che ne viene fatto.

In tabella sono indicate, procedendo dall'alto verso il basso, proprietà sempre più stringenti di precisione, affiancate ciascuna dalla descrizione di un caso tipico in cui è richiesta.

PROPRIETA'	CASO TIPICO
Attendibilità	Nella **messaggistica elettronica** le marche temporali sono generate per l'utente e non sono controllate dal sistema di calcolo.
Monotonicità	In un **sistema operativo**, in un **file system**, in un **database** le marche temporali devono consentire un preciso e controllato ordinamento temporale degli eventi.
Sincronismo	Nelle **applicazioni distribuite** i calcolatori coinvolti devono sincronizzare i loro orologi con marche temporali per poter prendere decisioni "allo stesso istante"
Precisione & inalterabilità	Nell'**autenticazione** di documenti informatici è indispensabile un'indicazione precisa e non alterabile del momento in cui è stata apposta la firma.

In questa sede interessa la sicurezza: esamineremo quindi dapprima come sia possibile ottenere una misura precisa ed oggettiva del tempo e poi come si possa impiegarla per attribuire una marca temporale **sicura** ad un documento che deve avere valore legale.

1 - Misura del tempo



La misura del tempo richiede:

1. un "generatore" di eventi periodici,
2. un "contatore" per numerarli,
3. un "visualizzatore" del valore raggiunto dal conteggio.

La sorgente degli eventi può essere:

- **naturale** (un fenomeno fisico come la frequenza di risonanza dell'atomo di Cesio 133 od un fenomeno astronomico come il passaggio del sole sul meridiano di Greenwich),
- **artificiale** (come un peso, una molla o un oscillatore al quarzo).

La precisione della misura dipende dalla precisione del generatore; è però sempre possibile migliorarla impiegando un secondo e più stabile generatore esterno per sincronizzare ogni tanto lo stato del contatore.

ESEMPI – Usuale è il sincronizzare i nostri orologi da polso sul segnale orario della RAI; molte sveglie si sincronizzano automaticamente su un segnale orario preciso che ricevono via radio.

La Comunità internazionale ha assunto come riferimento temporale primario il comportamento medio di 260 orologi atomici posti in Istituti di Metrologia di più di 40 nazioni (**TAI** o **Tempo Atomico Internazionale**).

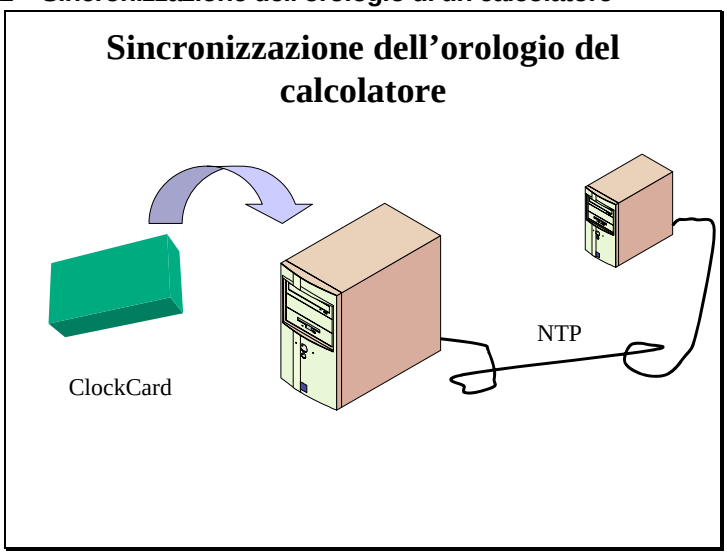
A tal fine ogni orologio atomico è connesso ad un server che ne conta i periodi di oscillazione; l'unicità della misura è ottenuta facendo interagire i server tramite una rete che li collega tutti.

Un secondo riferimento internazionale è fornito da **UTC** (**Tempo Universale Coordinato**) che apporta una piccola correzione alle misure del TAI: l'obiettivo è far avvenire esattamente alle 12:00:00 il passaggio del sole a Greewich.

Le misure TAI e UTC sono diffuse nei vari Paesi ed impiegate poi da Enti e da persone per sincronizzare orologi meno stabili e meno precisi. Il caso che qui interessa discutere è la sincronizzazione degli orologi di calcolatori impiegati nella generazione e nella verifica di firme digitali.

[65] Le considerazioni che seguono sono tratte da Riccardo G. Basile: "Relazione per Sistemi informativi II", 2002.

2 – Sincronizzazione dell'orologio di un calcolatore



L'inizio e la fine di ogni attività elementare del hardware di un calcolatore sono scanditi da un'onda quadra fornita da un oscillatore elettronico controllato a quarzo (il cosiddetto **clock**). Tutti i calcolatori impiegano il loro clock anche per incrementare appositi contatori HW e SW, da cui poi estraggono indicazioni temporali.

La qualità della misura del tempo non è in generale molto elevata:

- la **risoluzione** è di norma 1 msec,
- la **precisione** dipende da diversi fattori ambientali (temperatura, umidità, pressione, campi magnetici),
- l'**accuratezza** è piuttosto bassa (scarti anche di 20 secondi in un giorno),
- la **monotonicità** non è garantita (gli utenti possono *settare* l'orologio come vogliono).

Nelle applicazioni in cui è richiesta una buona precisione nella misura assoluta del tempo è dunque indispensabile ricorrere alla sincronizzazione, cosa che richiede l'aggiunta di HW e/o SW.

ESEMPI - Esistono in commercio schede hardware (ad es. la **ClockCard** della beaglesoftware.com) contenenti un orologio molto stabile con cui è periodicamente sincronizzato l'orologio locale del computer.

La sincronizzazione può essere fatta anche instaurando collegamenti, tramite la rete, con affidabili sorgenti remote. A tal fine esistono prodotti software sia commerciali, che free; tra questi ultimi molto apprezzato è **NTP**, un protocollo ed un daemon messi a punto all'Università del Delaware (www.ntp.org).

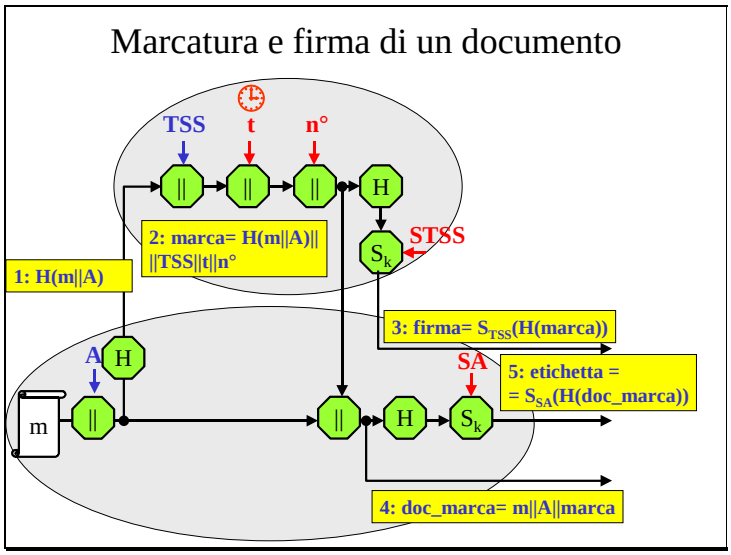
3 - Firma digitale con marca temporale

Un **preciso riferimento temporale** nella firma digitale di un documento consente a chi lo riceve sia di conoscere con esattezza la data di creazione, sia di accertare che a tale data la chiave di firma era valida[66].

Nel PGP la marca temporale è apposta direttamente dal programma del firmatario, essendo un presupposto che gli utenti devono fidarsi uno dell'altro. In caso contrario occorre prevedere che la marcatura temporale del documento sia svolta da un'**Entità diversa dal firmatario e dotata di un orologio sicuro**.

Un **servizio sicuro di marcatura temporale** deve avere diverse proprietà.

1. L'istante di marcatura di un documento deve essere quello indicato o dal TAI, o da UTC.
2. La marca temporale deve riferirsi univocamente ad un istante, ad un documento ed a chi l'ha firmato.
3. Chiunque deve poter accorgersi che una marca temporale è stata alterata.
4. Deve essere possibile far marcare temporalmente anche dati riservati.
5. Chiunque deve potere far marcare suoi documenti e verificare la marca apposta su documenti degli altri.



Una possibile organizzazione del servizio è indicata in figura.

1. L'utente **A**, prima di firmare un documento **m**, invia a **TSS** il suo identificativo e la sola impronta di **m**, per difenderne l'eventuale riservatezza.
2. **TSS** concatena **H(m)** con il suo ID, con l'indicazione temporale **t** e con il numero progressivo **n°** che ha attribuito alla marca.
3. **TSS** restituisce ad **A** sia questo dato in chiaro, sia la firma che ha apposto al tutto.
4. **A**, dopo aver verificato la correttezza della marca (l'impronta arrivata a **TSS** potrebbe essersi deteriorata lungo il percorso od aver subito un qualche attacco attivo), la allega al suo documento.
5. **A** firma l'intero documento ed allega la firma della marca apposta da **TSS**.

ESEMPIO – Il sito italiano di Timbrodigitale certifica a pagamento documenti con un'organizzazione di questo tipo.

[66] La legislazione italiana prevede che la vita massima di una chiave di firma sia di 3 anni.

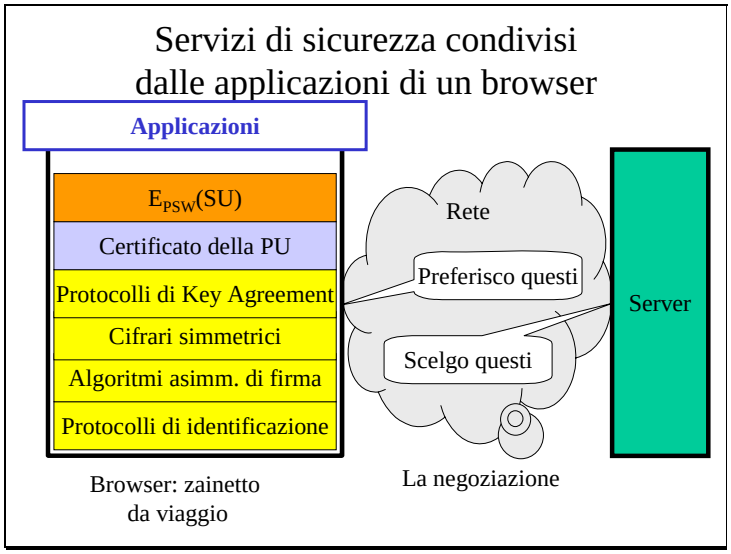
Per il buon funzionamento occorrono due presupposti:
1. nessuno deve riuscire a far marcare un documento con un tempo diverso da quello reale;
2. tutti devono poter verificare l'origine della marca temporale.

La soluzione usuale è prevedere che TSS sia un Ente **fidato** e che la chiave di verifica della sua firma sia **certificata**. Lo standard **PKIX** prevede che il servizio di time stamping sia incluso tra quelli offerti da una PKI.

Molti sono i dubbi sull'efficienza di questa soluzione a fronte del prevedibile crescere della domanda. Esiste anche il problema della chiave con cui TSS firma le marche: un gran numero di autenticazioni ne mette, infatti, in pericolo la segretezza, rendendo praticamente obbligatorio sia attribuirle una vita molto breve[67], sia adottare poi complesse procedure per convalidare la data di creazione di documenti a vita più lunga.

Sono in corso interessanti progetti di ricerca (v. ad es. "cuculus" della CA dell'Estonia: www.cyber.ee) per mettere a punto nuovi metodi di marcatura temporale che non richiedano autorità **assolutamente fidate**, siano immuni dal pericolo della "**rottura**" delle chiavi di firma, siano **scalabili** a grandi numeri di utenti e consentano azioni di **verifica off-line**.

6.2 Servizi sicuri per le comunicazioni in rete



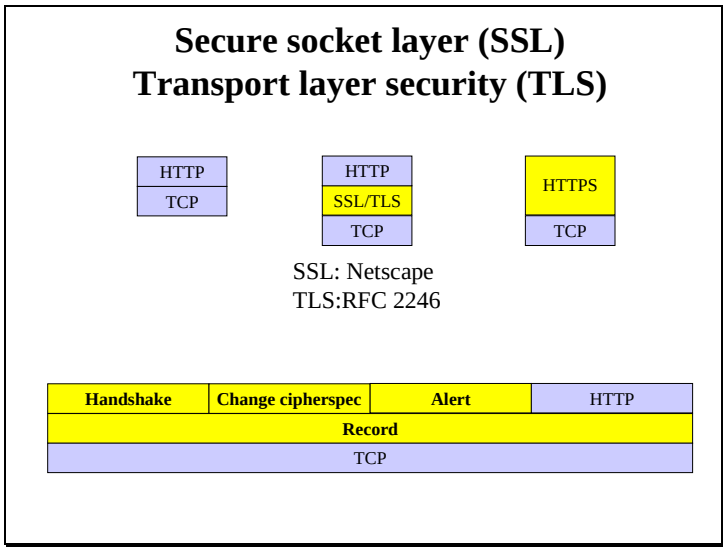
Ogni host ha in generale il compito di eseguire diverse e distinte **applicazioni distribuite**; ciascuna di queste ha a sua volta la necessità di comunicare in modo sicuro sulla rete e deve quindi potersi avvalere di meccanismi per l'autenticazione, l'identificazione e la riservatezza.

Un accorgimento che consente di ridurre notevolmente la complessità di progetto delle applicazioni è quello di predisporre al di sotto, e quindi a disposizione di tutte, un insieme **ridondante** di meccanismi sicuri (portachiavi, certificati, algoritmi di firma, di key agreement, di cifratura e d'identificazione).

In questo caso, una volta creata la connessione con un corrispondente, è necessario che le due parti aprano una fase di **negoziazione**, per arrivare a definire quali azioni svolgere e come svolgerle.

6.2.1 SSL/TLS

Il primo supporto sicuro per applicazioni Internet, denominato **SSL** (*Secure Sockets Layer*) e concepito per essere collocato tra il protocollo **HTTP** ed il sottostante strato di trasporto **TCP**, è stato realizzato da Netscape, che ne ha anche curato le prime tre versioni.



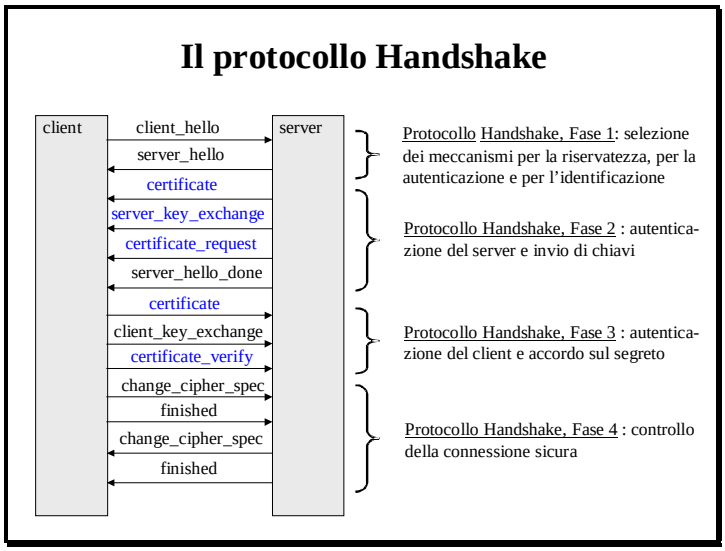
Lo standard Internet (RFC 2246) che discende da SSL è detto **TLS** (*Transport Layer Security*). La sigla HTTPS indica un'applicazione HTTP che si avvale dei servizi di sicurezza di TLS. Il protocollo SSL/TLS[68] è formato, in realtà, da quattro protocolli distinti:

- **Handshake** si occupa della negoziazione, dell'identificazione e del key agreement,
- **Alert** si occupa della gestione di tutte le segnalazioni di allarme,
- **Change cipherspec** consente di allineare le due parti sull'uso di una nuova *suite*,
- **Record** calcola/verifica le etichette di autenticazione e cifra/decifra i dati.

[67] La legislazione italiana prevede un mese.
[68] Una trattazione più dettagliata è presente in [6], cap 17.

1 - Il protocollo Handshake

Nella sottostante figura, procedendo dall'alto verso il basso, è evidenziata una tipica sequenza di messaggi (alcuni sono opzionali) che client e server si scambiano durante le quattro fasi del protocollo di **Handshake**.



Nella **Fase 1**, con il messaggio di **hello,** il client indica, in ordine di preferenza, i meccanismi di sicurezza di cui è dotato.

Tramite **server_hello**, il server comunica al client quali meccanismi ha scelto ed indica il metodo che dovranno impiegare per arrivare a condividere un **master_secret** (più avanti vedremo che è possibile avvalersi dello scambio DH o del Cifrario RSA).

Nella **Fase 2** il server invia il suo certificato X.509 e richiede, se vuole, il certificato del client.

Nella **Fase 3** il client invia, se richiesto, il suo certificato e lancia il procedimento di accordo sul segreto.

Nella **Fase 4** client e server si dichiarano pronti ad applicare ai dati quanto hanno in precedenza concordato e calcolato.

2 - La condivisione e l'impiego del master_secret

Abbiamo appena detto che il protocollo Handshake, all'inizio di ogni sessione di comunicazione, si occupa di far condividere un segreto alle due parti. Vediamolo più in dettaglio.

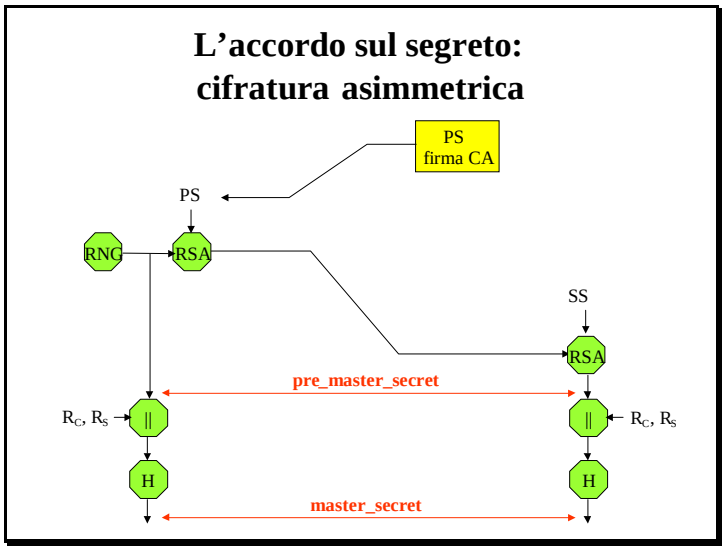
La prima cosa da chiarire è che in realtà di segreti ne servono complessivamente sei, tre per il flusso di dati che il client invia al server ed altri tre per il flusso nella direzione opposta. Per ogni direzione è, infatti, richiesto:

- un primo dato segreto per autenticare/verificare l'impronta di ogni blocco di dati scambiato,
- un secondo dato segreto per inizializzare il Cifrario simmetrico che cifra/decifra i blocchi (un vettore di inizializzazione se si usa la modalità CBC, un seme se si usa OFB o CFB),
- un terzo dato segreto per definire la chiave di sessione $\mathbf{k_s}$ del Cifrario a blocchi.

La seconda cosa da chiarire è che passare da uno a sei segreti non costituisce un problema. La stringa di bit del master_secret, lunga 48 byte, è impiegata come seme di una funzione pseudocasuale realizzata con hash MD5 e SHA-1: una serie di iterazione consente di ottenere i sei dati segreti con la lunghezza appropriata.

L'accordo sul master_secret parte da due numeri scelti a caso, $\mathbf{R_C}$ e $\mathbf{R_S}$, che client e server si scambiano in chiaro con i messaggi di **hello**: i due nonce costituiscono una difesa preventiva agli **attacchi di replica**.

Il protocollo mette a disposizione quattro modalità d'accordo.



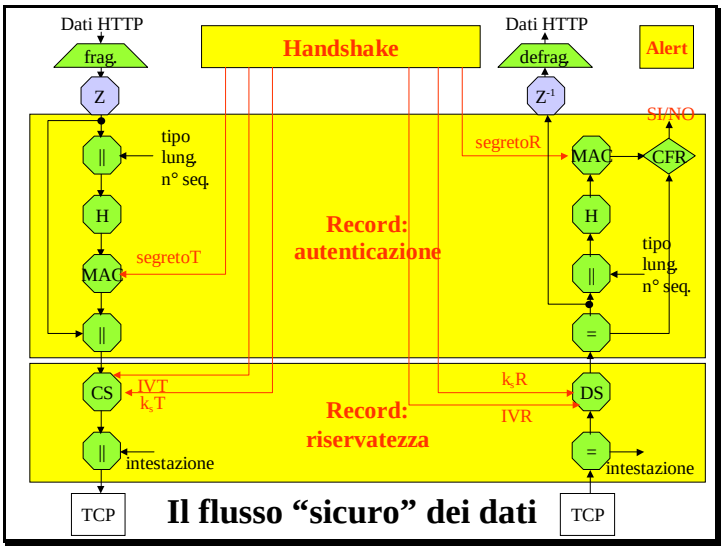
Le modalità **anonimous DH**, **fixed DH** ed **ephemeral DH** sono già state illustrate e commentate rispettivamente a pag. 78, a pag. 96 ed a pag. 97.

Un'ulteriore modalità di accordo sul master_secret prevede l'uso di **RSA**: in questo caso è il solo client a scegliere la chiave di sessione ed RSA gli consente di comunicarla al server in modo riservato. La chiave pubblica di cifratura è estratta dal certificato che il server ha precedentemente inviato al client.

Una volta calcolato il master_secret, prima il client e poi il server comunicano all'altro sia la volontà di mettere in linea quanto hanno appena concordato (messaggi di **change_cipher_spec** della Fase 4 di Handshake), sia gli estremi per controllare di aver calcolato lo stesso dato (messaggi di **finished**).

3 - Il Protocollo Record

Sul flusso dei dati opera il protocollo **Record**.



In figura sono indicate (rispettivamente a sinistra dall'alto verso il basso, a destra dal basso verso l'alto) le azioni che ciascuno dei due corrispondenti deve svolgere durante la trasmissione e la ricezione di un messaggio. I meccanismi per la sicurezza si avvalgono dei sei segreti che entrambi i corrispondenti hanno dedotto dal master_secret.

Trasmissione - Il messaggio è frammentato ed eventualmente compresso. Ogni pacchetto, secondo le decisioni prese da Handshake, può essere o non essere autenticato e cifrato. Al termine delle elaborazioni il pacchetto è completato con un'intestazione.

Ricezione - Su ogni pacchetto, una volta eliminata l'intestazione, è operata prima la decifrazione, poi la verifica dell'autenticità ed infine la decompressione e la ricomposizione.

6.2.2 IPv4

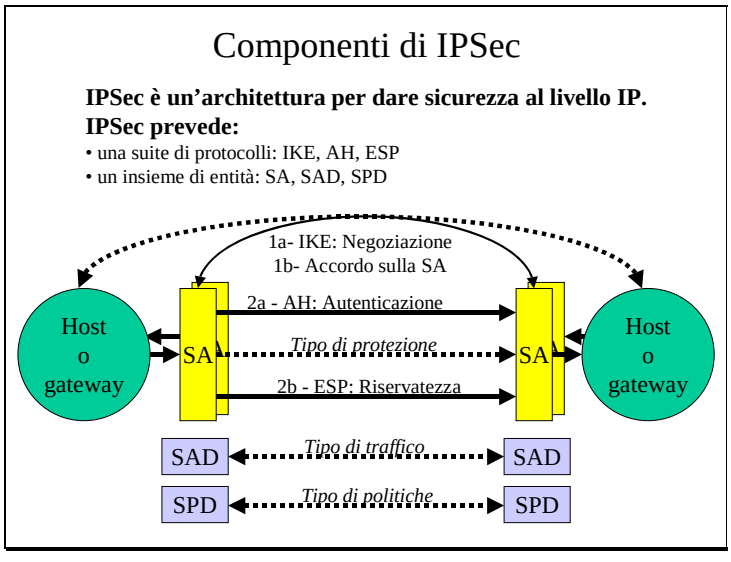
E' convinzione largamente condivisa che la versione attualmente in servizio dell'Internet Protocol (**IPv4** o semplicemente IP) non è adeguatamente sicura e non può quindi fronteggiare efficacemente il tasso di crescita di Internet e la continua richiesta di nuovi servizi.

La pressoché totale assenza di meccanismi di sicurezza in IPv4 discende dalla primitiva ipotesi che il protocollo poteva sia contare sulla collaborazione di tutti gli utenti, sia avvalersi di collegamenti di rete fisicamente sicuri. Tale ipotesi si è però dimostrata non realistica: le comunicazioni in transito sulla rete sono state, infatti, un facile bersaglio per diversi tipi di attacchi intenzionali.

- ***Packet sniffing***: ogni nodo posto tra mittente e destinatario può leggere i pacchetti in transito ed acquisire informazioni riservate.
- ***IP spoofing***: è possibile falsificare l'indirizzo del mittente apposto nell'intestazione di un pacchetto ed ingannare i servizi che autenticano in base a quel parametro.
- ***Connection hijacking***: un sabotatore può inserirsi in una comunicazione in corso introducendo dati errati o replicati.

Le attuali limitazioni di IPv4 non costituiscono un'imprevista novità: fin dall'inizio degli anni '90 ci si è posti il problema di realizzare un protocollo IP di nuova generazione in grado sia di prevenire illeciti monitoraggi/controlli del traffico, sia di offrire autenticazione/riservatezza alle comunicazioni tra ogni coppia di punti d'accesso alla rete.

6.2.3 IPSec



La definizione di un'**architettura sicura** per il livello IP (IP Security o **IPSec**), avviata nel 1994 con il rapporto *Security in the Internet Architecture* predisposto da IAB (RFC 1636), è stata poi precisata in diversi RFC.

IPSec costituisce oggi un importante riferimento non solo per Internet, ma anche per ogni altra rete di tipo WAN o di tipo LAN.

In figura è delineata la struttura logica che IPSec prevede di interporre tra due punti della rete che intendono corrispondere: a grandi linee si distinguono tre **protocolli** per svolgere altrettanti servizi sicuri e tre **strutture di dati** per istanziarli come desiderato dagli utenti.

Prima di scendere nel dettaglio è opportuno mettere subito in evidenza che questa architettura non ha più quindi la proprietà di **connectionless** inizialmente scelta per il TCP/IP.

I servizi previsti riguardano:

- **l'autenticazione** dell'origine dei singoli pacchetti,
- **la riservatezza** delle informazioni trasportate dai pacchetti,
- **la negoziazione degli algoritmi e delle chiavi** da impiegare alle due estremità del canale sicuro.

Come in SSL/TLS, un'importante caratteristica di IPSec è quella di lasciare comunque agli utenti un'ampia possibilità di scelta sugli algoritmi da impiegare nella protezione delle loro comunicazioni. La differenza tra le due soluzioni è che la prima accede ai servizi per la sicurezza tramite il meccanismo delle **API**, mentre la seconda ne affida completamente accesso e gestione al **kernel** del Sistema Operativo.

La collocazione di servizi sicuri a livello di rete consente il conseguimento di diversi obiettivi:

- garantisce un **utilizzo sicuro** della rete anche ad applicazioni sviluppate senza mettere in conto questo aspetto (connessioni remote di terminali, applicazioni client/server, posta elettronica, trasferimento di file, accesso al Web, e così via);
- impedisce agli intrusi di svolgere attacchi basati sull'**analisi del traffico**;
- impedisce l'arrivo di messaggi replicati alle macchine connesse in rete, evitando così fenomeni di congestione che possono arrivare fino alla **negazione** dei servizi per i quali sono state predisposte;
- offre una **sicurezza end-to-end** tra qualsiasi coppia di apparati di rete (host, router, firewall, gateway);
- protegge **in modo mirato** (e quindi anche differenziato) tutto il traffico a livello IP;
- consente ad Enti ed Organizzazioni di impiegare Internet sia per estemporanee esigenze di comunicazione (ad es. tra un dipendente fuori sede e la rete locale della sua azienda), sia per flussi permanenti di traffico tra sedi remote (di una stessa Organizzazione o di Organizzazioni strettamente cooperanti).

6.2.4 IPv6

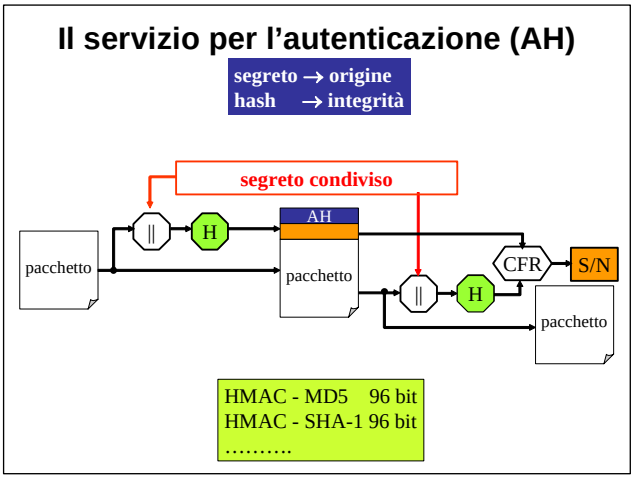
L'architettura **IPSec**, su alcuni punti ancora in corso di definizione, ha dato origine alla **versione 6** di **IP** (le prime specifiche sono del 1994). Dal 1996 esiste una rete (detta ***6Bone***), con un'estensione mondiale ed un'organizzazione gerarchica, dedicata alla sperimentazione di **IPv6**. L'obiettivo è duplice: verificare l'efficacia delle nuove protezioni ed individuare le modalità per fare coesistere nel medio termine il vecchio ed il nuovo protocollo.

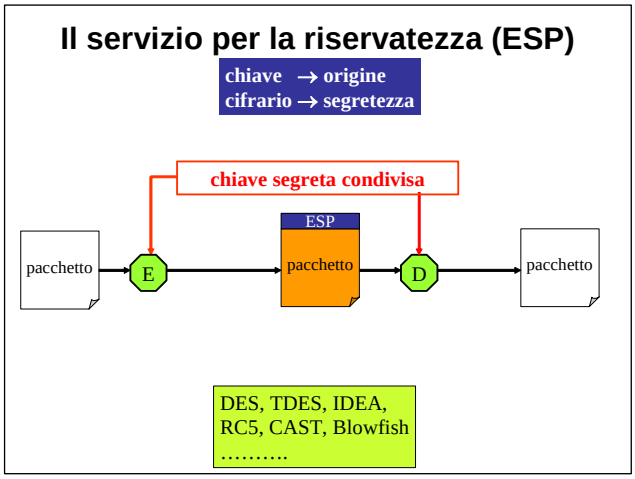
ESEMPIO - In Italia è possibile collegarsi a 6Bone attraverso due router **pTLA** (*pseudo top level aggregator)*, uno per utenti Telecom e l'altro per utenti di differenti *provider*.

L'intestazione di un pacchetto IPv6 è di **40 byte**, imposti prevalentemente dalla decisione di esprimere con 128 bit gli indirizzi della sorgente e della destinazione (IPv4 impiega solo 32 bit). Esistono anche campi prima non previsti: tra questi ci serve qui ricordare il campo ***Next Header***, impiegato per dichiarare quale servizio di sicurezza è stato applicato al pacchetto dalla sorgente e quale quindi dovrà essere completato, quando il pacchetto arriverà alla destinazione; i parametri del servizio sono poi precisati da un'intestazione successiva (***extension header***).

1 - AH e ESP

IPv6 fornisce **un servizio di autenticazione** ed un **servizio di cifratura**. I due servizi, di tipo *end-to-end*, sono in alternativa uno all'altro e contraddistinti da due differenti *extension header*.



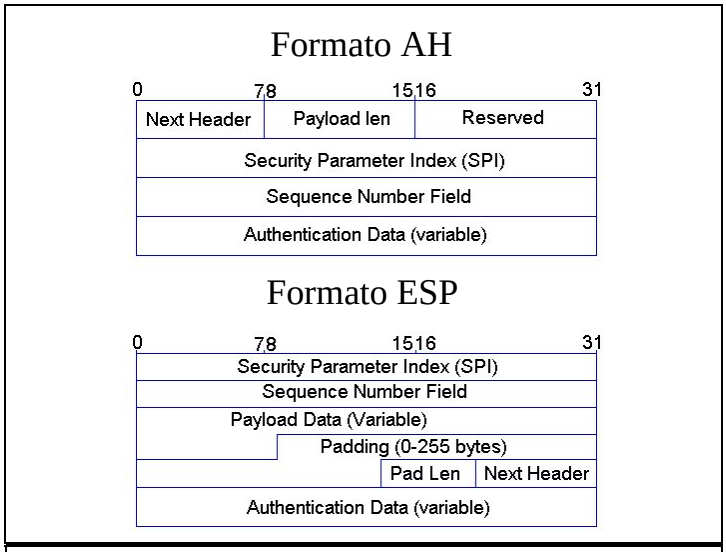


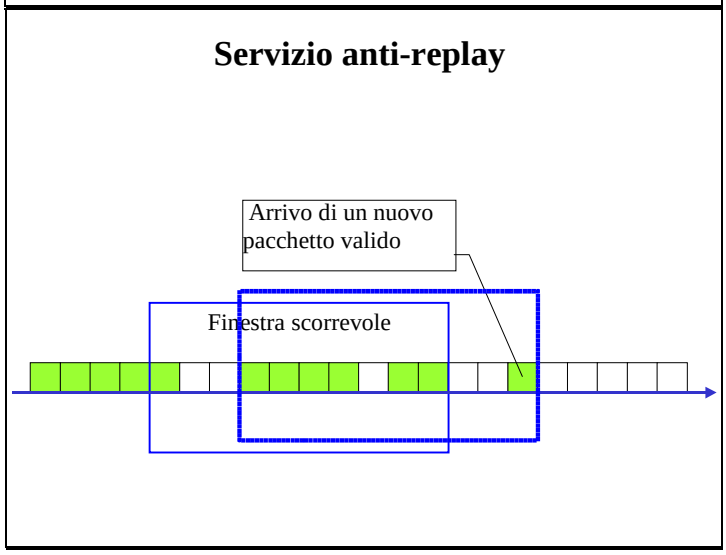
L'***Authentication header*** (**AH**) è apposto ai pacchetti su cui la sorgente ha operato per garantirne autenticità ed integrità; al suo arrivo indica alla destinazione di verificare che il pacchetto non sia stato modificato in transito o non sia stato generato da un intruso.

L'***Encrypted Security Payload header*** (**ESP**) è apposto ai pacchetti di dati che la sorgente ha sottoposto a cifratura; al suo arrivo chiama in causa la decifrazione e consente quindi solo alla legittima destinazione di rimettere i dati in chiaro.

Per svolgere tali servizi, sorgente e destinazione devono aver "stipulato un contratto" che sancisce quali protocolli, quali algoritmi e quali chiavi dovranno utilizzare per ogni "tipo" di traffico che intendono scambiarsi.

L'insieme di queste informazioni è detta **Associazione di Sicurezza** (***Security Association*** o **SA**), vale per un certo **tipo di traffico** e si riferisce ad una sola **direzione del flusso** di dati.





Sorgente e destinazione devono dunque mantenere in un proprio database (detto **SAD**) i parametri di tutte le associazioni di sicurezza che hanno negoziato (come, lo esamineremo più avanti).

Le intestazioni **AH** e **ESP**, anticipate da un Next Header rispettivamente di valore 51 e 50, hanno formati diversi.

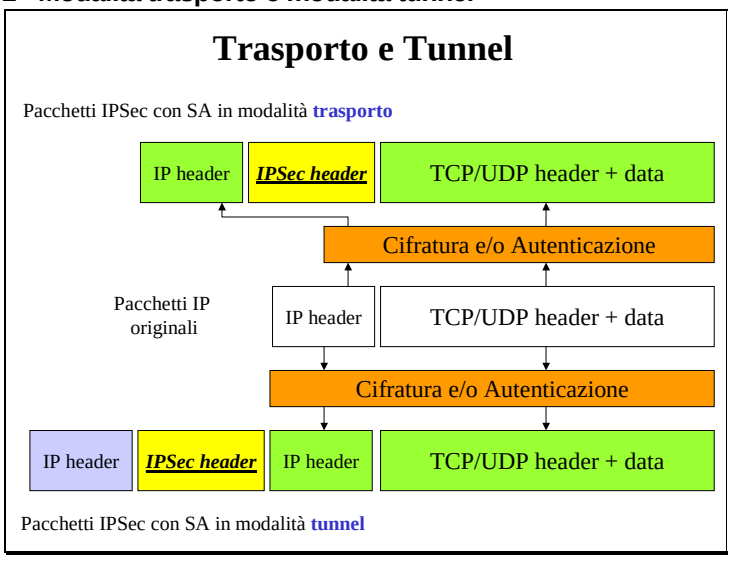
In entrambe i formati è previsto un campo **SPI** (*Security Parameter Index*) di 32 bit che serve alla destinazione per identificare l'associazione di sicurezza da usare.

Un ruolo importante per la sicurezza riveste anche il campo successivo (***Sequence number field***), in cui la sorgente indica il numero di sequenza attribuito al pacchetto per mettere in grado la destinazione di scartare ogni pacchetto replicato da un intruso.

Come è noto, IP non garantisce l'arrivo di **tutti** i pacchetti ed il rispetto del loro **ordine** originario. Per dare comunque alla destinazione la possibilità d'individuare pacchetti replicati, è usato il principio della "finestra scorrevole" illustrato in figura.

All'interno della finestra c'è posto per un certo numero di pacchetti consecutivi (la lunghezza di default della finestra è di 64 pacchetti). All'arrivo di un nuovo pacchetto si verifica la sua integrità (tramite il campo ***Authentication Data***) ed, in caso affermativo, si confronta il suo numero di sequenza con quelli contenuti nella finestra: se è tra quelli già convalidati o se ha un numero di sequenza inferiore, il pacchetto è scartato; se non è presente, la posizione è marcata e fatta coincidere con l'estremità destra della finestra.

2 - Modalità trasporto e modalità tunnel



Durante la definizione della SA è possibile scegliere tra due differenti modalità di trasporto dei pacchetti: il ***transport mode*** ed il ***tunnel mode***. Con entrambe le modalità, il servizio può poi essere o AH o ESP.

La differenza tra queste due modalità sta nel tipo di trattamento di sicurezza applicato al pacchetto.

Nel ***transport mode*** è cifrato e/o autenticato il *payload* del pacchetto IP e sono autenticate porzioni del suo header. La modalità trasporto è tipicamente impiegata, quando i due ***end point*** sono degli ***host***.

Nel ***tunnel mode*** l'intero pacchetto è cifrato e/o autenticato ed è poi incapsulato come payload in un altro pacchetto: in questo modo la sua origine, la sua destinazione ed il suo contenuto sono verificati (e decifrati nel

caso ESP) solo alle estremità del percorso. La modalità tunnel è tipicamente impiegata, quando i due host appartengono a reti locali diverse e quindi quando i due end point della inter-rete sono ***gateway*** (firewall o router).

Con la modalità tunnel è possibile impiegare Internet come supporto di una **rete virtuale privata**: se i pacchetti hanno l'intestazione ESP nessuno può vederne indirizzi e contenuto; è comunque in ogni caso impossibile modificarli o falsificarli senza che il ricevente se ne accorga.

3 - IKE, ISAKMP e Oakley

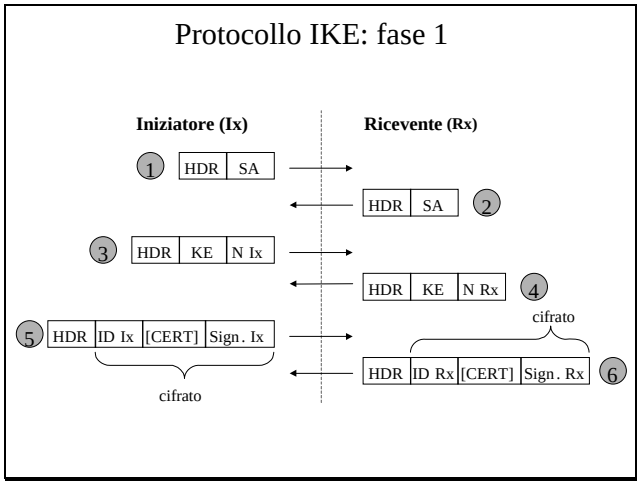
La negoziazione tra le due parti degli attributi di sicurezza delle loro SA è affidata al protocollo **IKE**, che a sua volta si basa sul *framework* **ISAKMP**(*Internet Security Association and Key Managment Protocol*) e su due protocolli di *key management*: **Oakley** e **Skeme**.

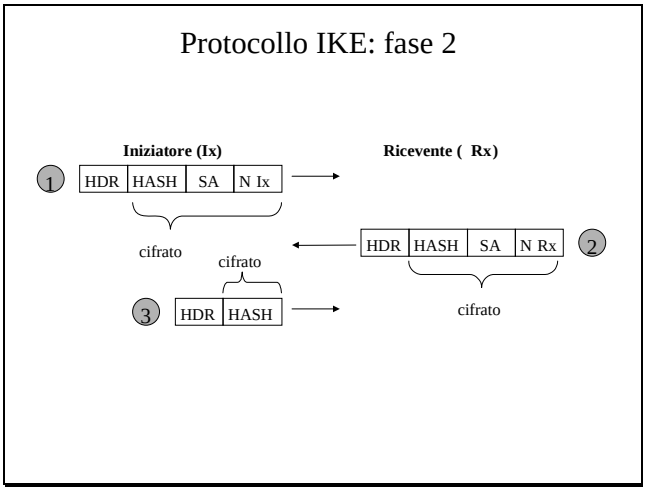
L'argomento è piuttosto complesso ed esistono tra l'altro diverse interpretazioni di quanto è stato specificato negli RFC, al punto che è in corso una revisione anche abbastanza profonda del protocollo (IKEv2): in questa sede ci limitiamo dunque ad uno stringato riassunto.

IKE stabilisce le procedure che consentono ai due *peer* di instaurare, modificare e cancellare SA.

ISAKMP definisce la struttura dei messaggi che si possono scambiare e le transizioni di stato che si dovranno eseguire per giungere alla creazione di un connessione sicura. La comunicazione dei messaggi ISAKMP è affidata ad un protocollo del livello di trasporto.

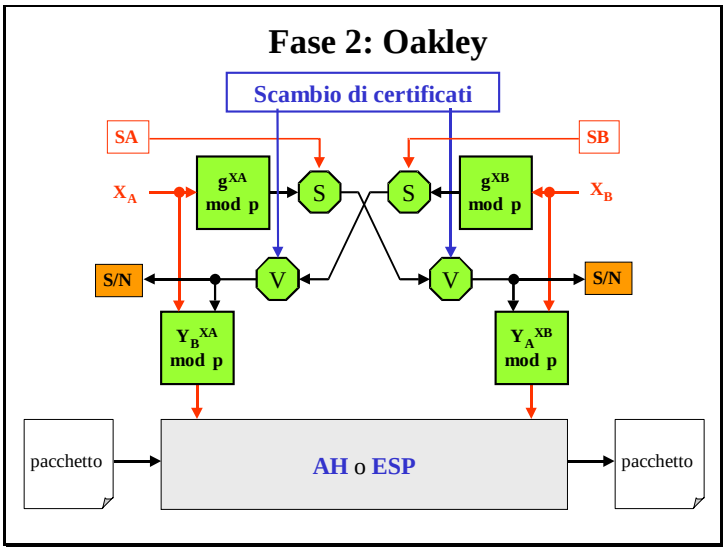
Per arrivare a definire i parametri di sicurezza di una SA, IKE si avvale tipicamente di due fasi, come è già stato detto a pag. 93. Nelle figure sottostanti sono riproposte, in forma diversa, le attività previste.





La **fase 1** si occupa dell'accordo sulla SA iniziale, o **ISAKMP SA**, garantendo l'identificazione mutua dei due peer e la protezione della loro identità a fronte di intercettazioni indebite.

La **fase 2** usa la chiave ottenuta al termine della fase 1 per stabilire in modo riservato i parametri di sicurezza di ogni SA (**AH SA** o **ESP SA**) che occorrerà successivamente instaurare per gestire nel modo negoziato i differenti flussi di dati.



Oakley è l'attuale scelta di default di IKE. In figura è mostrato come opera nella fase 2 al fine di far condividere alle due parti i segreti che dovranno poi impiegare nel traffico protetto o con il protocollo AH, o con il protocollo ESP.

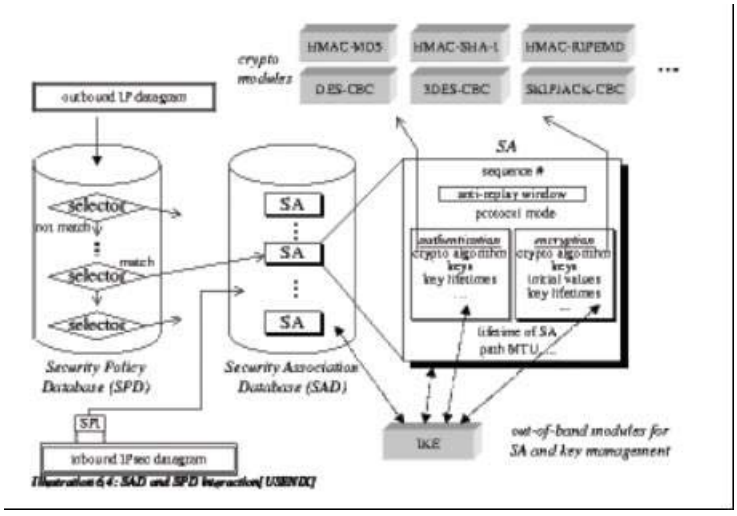
L'accordo sulle chiavi di una SA è dunque basato sulla versione ephemeral dell'algoritmo di Diffie-Hellman, ma sono lasciati molti gradi di libertà agli utenti (scelta dei parametri del gruppo moltiplicativo, del formato delle chiavi, dell'algoritmo di firma, del tipo di certificato).

Un importante accorgimento adottato nella fase 1 è lo scambio preliminare di due numeri a caso (detti *cookie*), che devono poi essere rinviati al mittente per dare inizio al vero e proprio protocollo.

Ciò serve a contrastare il ***clogging***, una particolare forma di **denial of service attack** con cui un avversario, dopo aver falsificato l'indirizzo del mittente (cosa relativamente facile), richiede continuamente alla vittima il calcolo di nuove chiavi di sessione obbligandola a fare elaborazioni onerose ed inutili. La difesa è basata sul fatto che è invece difficile per l'aggressore intercettare il dato di risposta inviato all'indirizzo falsificato.

4 – SAD e SPD

In figura, come riassunto di quanto abbiamo detto, è rappresentata l'organizzazione logica di un punto d'accesso alla rete gestito con IPSec.



Tutte le **SA attive** su una connessione sono contenute nel **SA Database** (SAD), mentre le politiche di sicurezza sono contenute nel **Security Policy Database** (SPD).

SAD descrive **cosa si vuole fare**: ad esempio fissa SPI, indica l'algoritmo crittografico da impiegare, memorizza il sequence number, fissa le dimensioni della finestra anti-replay, ecc..

SPD, tramite "selettori", descrive **come si vuole fare**: ad esempio da quali host si può ricevere pacchetti, quale traffico deve essere scartato, quale traffico deve essere cifrato, quale traffico deve essere autenticato, ecc..

Appositi comandi consentono di creare, cancellare i due database e di visualizzarne il contenuto durante il servizio.

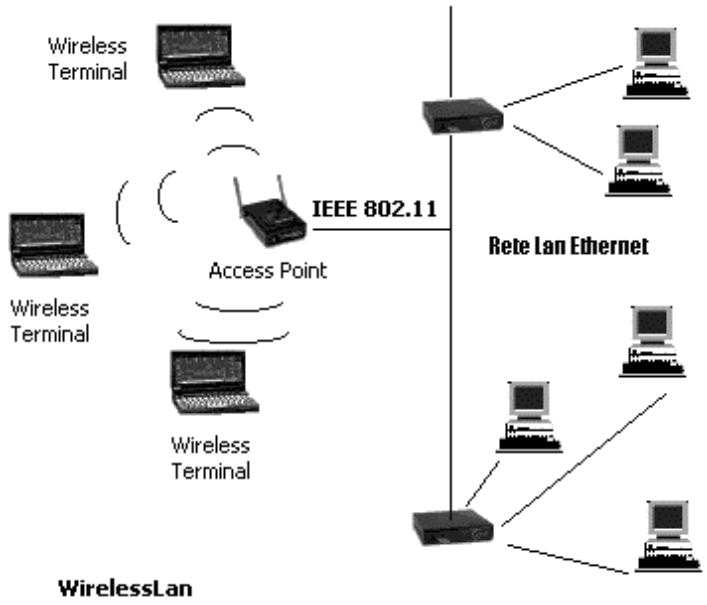
6.2.5 Reti wireless

Con **mobile computing** si intende un modello di elaborazione distribuita all'interno del quale gli utenti (non necessariamente solo umani) ottengono e restituiscono informazioni **dove**, **quando** e **come** vogliono.

La prima realizzazione di questo modello è stata la rete **wireless**, o **Wi-Fi**, introdotta con l'obiettivo principale di fare (o di ampliare) reti locali senza posare cavi. Il secondo obiettivo è stato quello di consentire agli utenti una qualche forma di mobilità.

Entrambi gli obiettivi sono conseguibili affidando ad onde radio la trasmissione dell'informazione[69].

Questa soluzione, adottata inizialmente dai personal computer Apple alla fine degli anni '90, ha poi coinvolto tutta l'industria ICT (al punto che l'unità per la rice/trasmissione di segnali radio è stata poi integrata all'interno dei processori) ed è oggi impiegata in milioni di case, di uffici e di luoghi pubblici.



In una rete wireless si distinguono due categorie di dispositivi: i terminali d'utente **WT**, predisposti per comunicare via radio, e gli *access point* **AP**, che interfacciano l'ambiente senza fili ad una rete cablata.

I WT possono o colloquiare direttamente tra loro (**modalità ad-hoc**) o collegarsi prima agli AP (in modo analogo al *roaming* della telefonia cellulare) e poi comunicare o con i server della rete fissa, o con altri WT (**modalità infrastruttura**). La "copertura" di un AP dipende dalla potenza di emissione e dalla frequenza di trasmissione: tipicamente può variare da qualche decina di metri in interno fino a qualche centinaio di metri in esterno.

Gli standard 802.11x, stabiliti da IEEE (*Institute of Electrical and Electronics Engineers*), definiscono le specifiche per i livelli *fisico* e *data-link* delle reti wireless, corrispondenti rispettivamente ai livelli 1 e 2 di OSI.

[69] Occorre però rispettare le norme di impiego delle frequenze radio stabilite in sede internazionale ed in sede nazionale; le regole italiane sono sancite dal DPR 447/2001

Il secondo standard in ordine di tempo, **802.11b** del 1997, ha avuto ed ha tuttora una diffusione enorme, essendo stato adottato da quasi tutte le grandi industrie del settore (IBM, Cisco, 3Com, ecc.).

Sono però anche emersi aspetti di inefficienza e, soprattutto, **punti di vulnerabilità** che hanno portato dapprima alla definizione di **estensioni compatibili** (**802.11a** e **802.11g**) e poi all'attuale standard **802.11i**.

Nel seguito cercheremo di dare almeno una prima idea sui principali accorgimenti per la sicurezza adottati da questi standard. Attenzione: non sono certamente gli unici a cui deve fare riferimento chi sviluppa applicazioni.

Esistono già ora, ed il loro numero continuerà ad aumentare, altri importanti **standard de facto** dedicati a supportare applicazioni innovative del mobile computing e/o ad ottenere una sempre più stretta integrazione con i servizi della video-telefonia cellulare.

ESEMPI - **Wimax**, **Zigbee**, **Ultra Wideband** sono le sigle di nuovi standard di comunicazione digitale via radio che sono attualmente oggetto di attenta considerazione in tutto il mondo.

Una citazione particolare merita la tecnologia messa a punto dalla Ericcson a partire dal 1994 e denominata **Bluetooth** (dal nome di un re vichingo del X secolo); attualmente sono moltissime le industrie che si sono unite alla Ericcson per creare e promuovere questo standard come soluzione globale per le comunicazioni wireless. Inizialmente è stata impiegata la frequenza di 1 Mbps per trasmettere dati tra apparati distanti pochi metri; oggi sono in uso frequenze più alte (fino a 3 Mbps) e si riescono a coprire distanze fino a 100 metri.

Le applicazioni iniziali hanno riguardato gli auricolari dei telefonini ed i computer indossabili; oggi sono le più svariate e centinaia di milioni di dispositivi incorporano ogni anno questa tecnologia.

1 - WEP[70]

In una rete wireless occorrono **meccanismi di sicurezza a livello fisico**: chiunque si trovi nell'area di copertura di un AP può, infatti, attuare agevolmente attacchi passivi ed attivi.

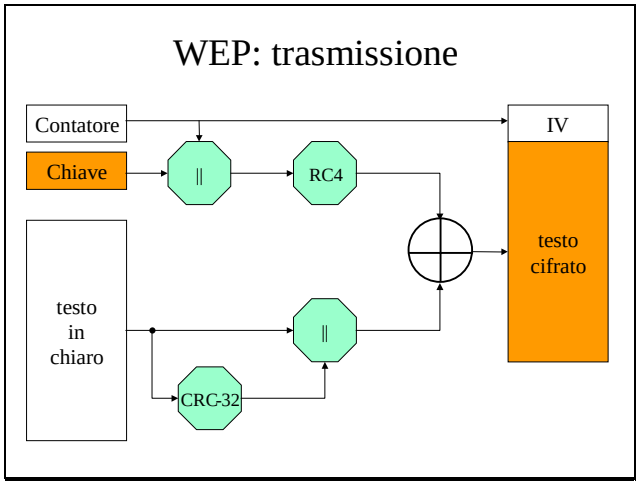
WEP (*Wired Equivalent Privacy)* è l'algoritmo appositamente creato per dare ad una rete locale wireless almeno lo stesso livello di sicurezza offerto dalle normali LAN.

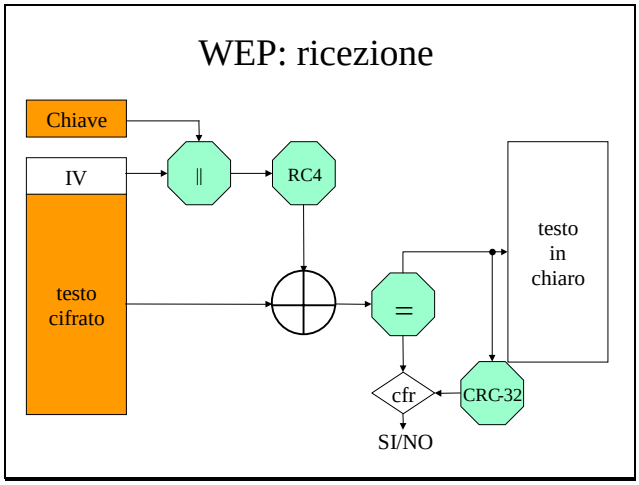
WEP impiega un **cifratura a flusso** per proteggere la riservatezza dei messaggi; a tal fine è utilizzato il Cifrario a flusso sincrono RC4 (v. pag. 66). Il **seed** è ottenuto concatenando

- un **vettore di inizializzazione** IV a 24 bit (modificato ad ogni trasmissione, di norma tramite un contatore, e trasmesso in chiaro in testa al cifrato)
- una **chiave segreta** (inizialmente di soli 40 bit, per via delle leggi statunitensi che vietavano l'esportazione di meccanismi crittografici "forti", ed oggi di 104 bit).

Per garantire l'integrità dei dati, ogni messaggio da trasmettere, prima della cifratura, è concatenato con un'etichetta calcolata dalla funzione **hash semplice CRC-32**.

Le sottostanti figure evidenziano il flusso dei dati in trasmissione ed in ricezione.





In questa realizzazione sono presenti diversi punti di vulnerabilità.

La scelta più debole è stata quella di far condividere la **stessa chiave** segreta a tutti gli apparati di rete e di installarla manualmente al loro interno: ciò determina il fatto che in pratica la chiave non viene mai cambiata e che tutto il traffico intercettato può essere utilizzato da un intruso per individuare il segreto e per spacciarsi da WT legittimo.

[70] Le considerazioni che seguono sono tratte dalle relazioni d'esame di Riccardo Tampieri e di Mirko Matoffi.

Si noti che se non si modifica la chiave, dopo 2^{24} messaggi il flusso di bit generati da RC4 si ripete. Su questo fatto può essere svolto un attacco, dato che due messaggi cifrati dalla stessa chiave presentano le stesse differenze che è possibile riscontrare sui testi cifrati:

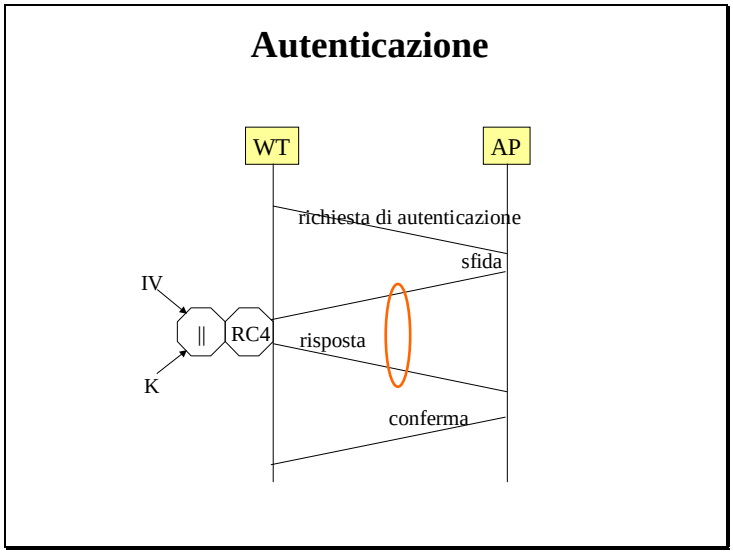
m1 ⊕ m2 = c1 ⊕ c2.

Un'ulteriore debolezza è costituita dal metodo di calcolo del seed: è stato dimostrato, infatti, che ciò consente di individuare la chiave con un attacco con solo testo cifrato[71], a patto di averne intercettato a sufficienza.

L'uso di CRC-32 rende infine possibile modificare i messaggi senza far apparire una violazione d'integrità.

WEP prevede anche un **protocollo a sfida/risposta** per consentire agli AP l'identificazione dei WT.

Il meccanismo impiegato nel protocollo è ancora il cifrario RC4, che il WT deve impiegare per cifrare la sfida con l'unica chiave di cui dispone.



I punti deboli sono due.

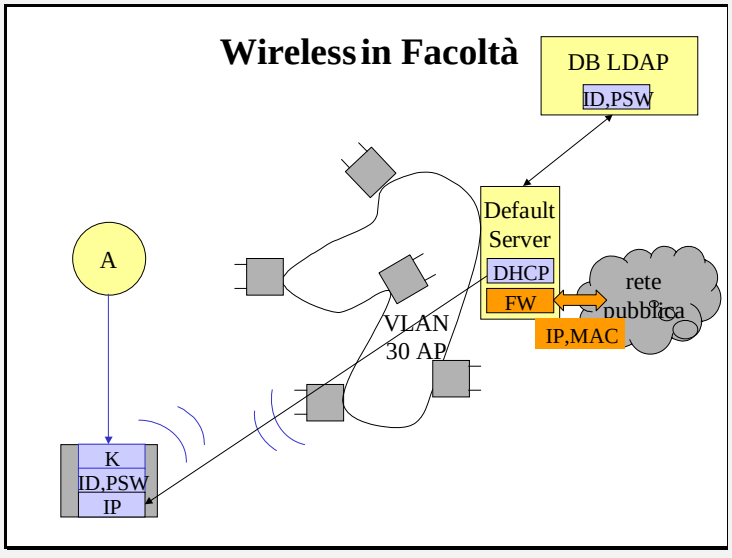
- L'identificazione è solo unilaterale, a fronte del fatto che è relativamente facile riuscire a spacciarsi da AP e portare quindi un attacco da "uomo in mezzo";
- L'uso della stessa chiave per l'autenticazione e per la riservatezza consente all'intruso di procurarsi prima tutto il testo cifrato che gli serve per individuarla e poi di impiegarla per farsi identificare.

L'impiego di un Cifrario a flusso è comunque sbagliato per definizione. Se uno è riuscito ad intercettare una sfida **s1** e la corrispondente risposta **r1 = IV || (s1 ⊕ k1)**, ottiene agevolmente **k1 = s1 ⊕ r1**.

Quando verrà a sua volta sfidato da un AT con **s2**, potrà dare una risposta corretta:

r2 = IV || (s2 ⊕ k1).

ESEMPIO – In figura è schematizzata la rete wireless e fissa installata in Facoltà nel 2002 per valutare i costi e le prestazioni della tecnologia WEP.



Un Access Point è costato circa 200 $, la scheda di rete da inserire nei PC circa 100 $.

Ogni utente è stato dotato di ID e di PSW.
I dati sono conservati in un Server database, a cui è attribuito il compito di controllare gli accessi alla rete di Facoltà e quindi a Internet.
Il protocollo di accesso è LDAP.

Una volta creata la connessione WT-AP, viene coinvolto il Default Server che richiede al Server di autenticazione il controllo della password.
Se il controllo è superato e se l'utente ha il diritto di accesso a Internet, il Server, tramite il Protocollo DHCP, gli assegna un indirizzo IP e gli consente di superare il Firewall FW.

2 - TKIP

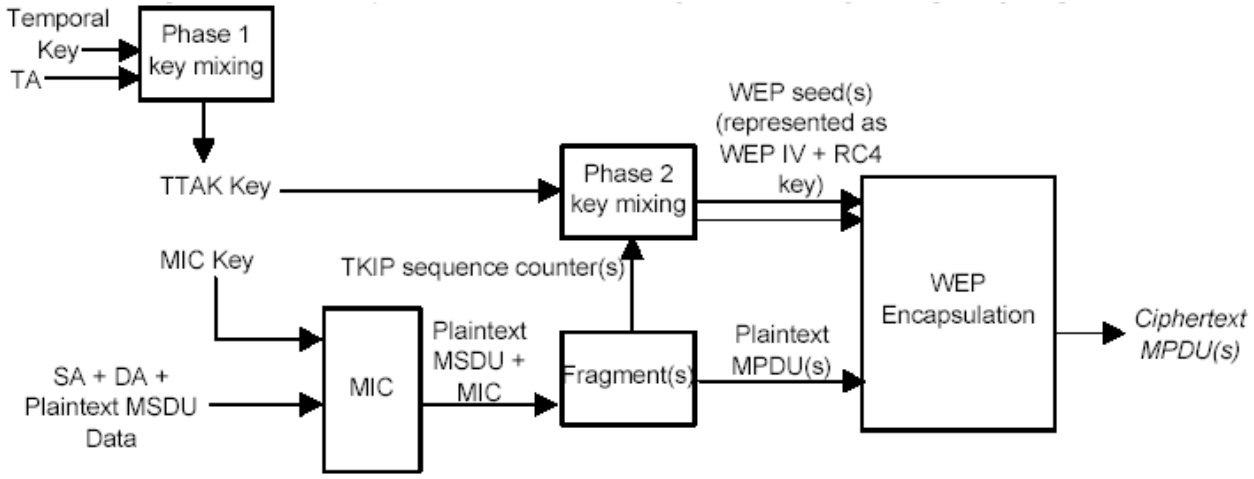
TKIP (*Temporal Key Integrity Protocol*) è un protocollo ideato per risolvere molte delle vulnerabilità del WEP, senza dover sostituire gli apparati che lo impiegavano: TKIP è dunque un aggiornamento esclusivamente software (o firmware), ma questo limita anche il livello di sicurezza ottenibile.

Tecnicamente TKIP è un *wrapper* del WEP, cioè un sistema che racchiude al proprio interno il protocollo WEP. Quattro sono gli accorgimenti introdotti per aumentare la sicurezza[72].

[71] L'attacco messo a punto da Fluhrer, Mantin e Shamir è descritto in http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf.
[72] v. http://www.cedar.intel.com/media/pdf/security/80211_part2.pdf.

1. **Protezione da messaggi falsi:** ad ogni messaggio in chiaro è aggiunto un MIC[73], calcolato da un cifrario simmetrico a blocchi in modalità CBC.
2. **Protezione dalla replica:** ad ogni pacchetto è aggiunto un numero progressivo precifrato.
3. **Protezione della chiave:** è impiegata la tecnica detta *key mixing*, che opera in due fasi. Nella Phase 1 una chiave detta Temporal Key è combinata con l'indirizzo MAC del dispositivo per ottenere una chiave intermedia (ciò evita che tutti i dispositivi della rete utilizzino la stessa chiave). Nella Phase 2 si utilizza un cifrario a blocchi per cifrare i numeri progressivi dei pacchetti utilizzando la chiave intermedia: il risultato è formato da 128 bit, in cui i primi 24 sono utilizzati come vettore IV, i restanti 104 come chiave del WEP.
4. **Protezione dal riutilizzo del flusso di chiave:** la tecnica detta *rekeying* sfrutta un frequente cambiamento della Temporal Key per evitare che TKIP utilizzi più di una volta lo stesso flusso di chiave. La distribuzione della Temporal Key deve avvenire in maniera sicura: di norma è impiegato o un server Kerberos o un server RADIUS).

La seguente figura (presa dal documento citato in nota 69) mostra il processo di cifratura:



La robustezza crittografica di TKIP, pur superiore a quella del WEP, non è particolarmente elevata e l'adozione di TKIP è stata quindi vista come un provvedimento valido solo per il medio termine. Il pregio maggiore di questo protocollo è stato quello di consentire l'impiego senza modifiche dell'hardware dei dispositivi WEP installati in precedenza.

3 - WPA

WPA (**Wi-Fi** *Protected Access*) è la prima realizzazione dello standard **802.11i** per quanto riguarda le porzioni applicabili ai dispositivi già esistenti: il rilascio di WPA è avvenuto in anticipo rispetto all'intero standard per poter mettere una pezza ai gravi problemi di sicurezza manifestati dal WEP.

WPA è stato progettato per essere un aggiornamento software o firmware, ma per metterlo in servizio occorre procedere all'aggiornamento di tutti i dispositivi di rete (access point e schede wireless).

I miglioramenti rispetto al WEP sono due: la **gestione delle chiavi** e l'**autenticazione**.

La gestione delle chiavi si appoggia al protocollo TKIP: ogni utente dispone di una propria Master Key, che è utilizzata da TKIP per generare la chiave da usare per la cifratura dei dati. La chiave di cifratura è usata solo una volta ed è poi cambiata automaticamente; ciò riduce notevolmente la possibilità che il sistema sia compromesso.

L'autenticazione si basa sul protocollo **EAP** (*Extensible Authentication Protocol)*, che prevede un **Authentication server** dotato di un database in cui conserva le coppie user name/Master-Key: prima di accedere liberamente alla rete un client deve essere autenticato dal server e viceversa (autenticazione reciproca). Il protocollo EAP è a sfida/risposta, impiega quattro tipi di pacchetto (**request**, **response**, **success** e **failure**) ed è in grado di supportare diversi meccanismi di autenticazione (smart card, Kerberos, TLS, ecc.), cosa che elimina la necessità di negoziazioni preventive.

WPA non supporta metodi di protezione per le reti ad-hoc ed algoritmi crittografici veramente robusti.

Queste proprietà sono invece previste nella versione finale di 802.11i. Come WPA, anche questo standard si appoggia a TKIP per la gestione delle chiavi, ma utilizza AES come algoritmo di cifratura. La maggiore complessità degli algoritmi richiederà hardware più potente, rendendo 802.11i incompatibile con i dispositivi aderenti allo standard 802.11b.

[73] Nella terminologia delle reti wireless si preferisce usare la sigla MIC (Message Integrity Code) per non creare confusione con il protocollo MAC (Media Access Control) dello strato di collegamento.