View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AMS Tesi di Laurea

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

CRITTOGRAFIA ED AUTENTICAZIONE NEL SISTEMA KERBEROS

Tesi di Laurea in Algoritmi della Teoria dei Numeri e Crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Francesca Corni

I Sessione
Anno Accademico 2012/2013

A chi mi è stato vicino, mio malgrado,
a chi crede in me,
a chi ho trascurato.

Indice

Introduzione

Tutti gli strumenti crittografici sono ideati per garantire la sicurezza di computer e comunicazioni. Gli algoritmi crittografici però non sono in grado, da soli, di risolvere tutti i problemi; ad esempio occorre una procedura più complessa per affrontare la distribuzione di chiavi pubbliche. Nei protocolli a chiave pubblica Alice possiede due chiavi, una pubblica ed una segreta. Quella pubblica viene diffusa e sarà utilizzata da Bob (o da chi altri vorrà mandare messaggi ad Alice) per cifrare il contenuto del suo messaggio. Come può però Bob fidarsi dell'identità di Alice? Potrebbe esserci un intruso, convenzionalmente chiamato Eva, che, spacciandosi per Alice, la impersona e diffonde la propria chiave. Questo è un classico problema di autenticazione.

Incontriamo tutti i giorni problemi di questo tipo e sempre con maggior frequenza nella nostra società, dove sono sempre più diffusi contatti e rapporti, anche commerciali, via rete: se voglio, ad esempio, acquistare qualcosa presso un negozio elettronico, dovrò possedere una prova del fatto che sia un negozio autentico. I protocolli per la sicurezza combinano generalmente algoritmi crittografici differenti e, per questo, sono più complessi degli algoritmi stessi. Il problema dell'autenticazione è dunque molto importante nelle comunicazioni informatiche. Alice, per dimostrare a Bob la propria identità, può utilizzare varie informazioni che la caratterizzano:

- qualcosa che lei sa (password, chiavi segrete ...);
- qualcosa che lei ha (ad esempio una smart-card);
- qualcosa che lei è (attraverso dispositivi che controllano le impronte digitali, la retina dell'occhio o misurano il tempo di battitura, ad esempio);
- dove lei si trova (non è però un'informazione sufficiente in quanto può accadere che non solo Alice si trovi in quel luogo fisico);

Alcuni metodi, a prima vista, sembrano efficaci e inattaccabili, come il confronto di impronte digitali, ma possiamo individuare in pochi minuti delle falle: ad esempio le impronte

potrebbero essere falsificate, non sono stabili per i bambini ...
I metodi che cercano di sfruttare ciò che una persona è non sono dunque i più usati perchè, a livello informatico-tecnologico, non sono così determinanti le caratteristiche che identificano un individuo. Gli scenari più comuni di autenticazione sono dunque basati sulla conoscenza di una **chiave**, pubblica o segreta. In particolare le situazioni più diffuse prevedono utenti, o entità, che condividono una password tra loro, utenti che condividono una password con un'autorità fidata o utenti che dispongono di una chiave pubblica.

Kerberos è un servizio di mutua autenticazione che raggiunge tre obiettivi:

- è sicuro: nessuna password viene trasmessa in chiaro;
- attraverso l'esistenza di un unico insieme di credenziali, permette di accedere a tutti i servizi distribuiti sulla rete;
- è basato sulla fiducia che gli utenti ed i server ripongono in una terza parte: un server di autenticazione centralizzato;

Kerberos utilizza i "*tickets*"; ogni ticket è un messaggio crittografico la cui validità è limitata nel tempo, e serve a provare l'identità di un utente ad un server, senza bisogno di inviare password in chiaro, e senza che sia necessario memorizzare la password sulla macchina dell'utente.

Lo scopo del sistema Kerberos è fornire accesso sicuro alle risorse distribuite su una rete insicura. Due possibili soluzioni sono:

- attraverso la crittografia a chiave pubblica, che richiede una infrastruttura, ad esempio l'uso di certificati digitali;
- mediante la crittografia simmetrica; l'idea di base è quella dal protocollo di Needham-Schroeder: una TTP (Trusted Third Party), di cui si fidano due entità, che chiameremo A e B, condivide con ciascuno degli utenti una chiave permanente. Grazie a queste chiavi, ed alla fiducia che A e B ripongono in TTP, la terza parte genera una chiave di sessione, condivisa tra A e B, attraverso la quale i due possono dialogare in maniera sicura. È la strategia impiegata da Kerberos.

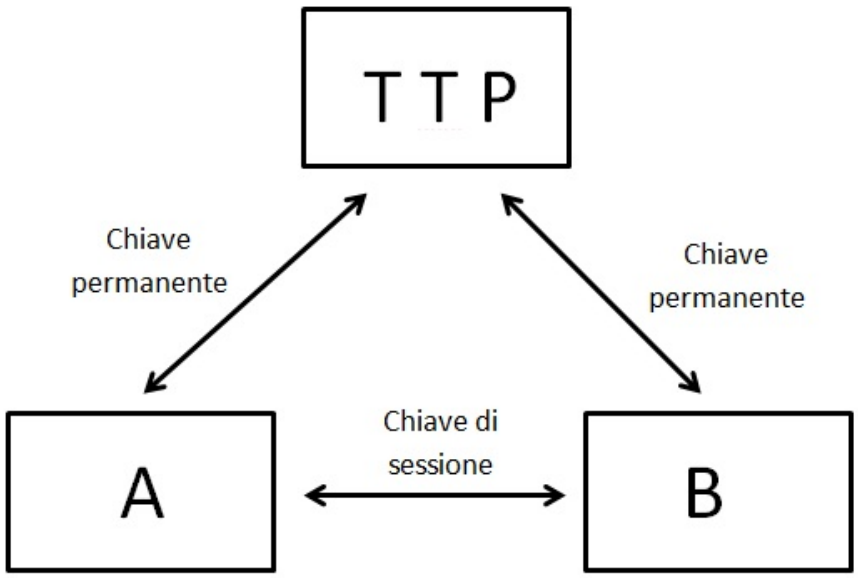

Figura 1: Schema semplificato del protocollo di Needham-Schroeder

Capitolo 1

Struttura fondamentale di Kerberos

1.1 Prima di Kerberos

Lo standard di autenticazione precedente Kerberos si chiamava *autenticazione per asserzione*. Quando un utente esegue un programma che accede ad un servizio in rete, e questo programma prende il nome di *client*, il client asserisce al servizio (*server*) di essere in esecuzione su richiesta di quel determinato utente. In pratica client e server si scambiano un'informazione segreta, di solito una password, conosciuta da entrambi. Questo sistema non garantisce l'identità dell'utente che accede al servizio client, quindi, anche se ha successo la fase di scambio del segreto, è possibile che Eva si sia sostituita ad Alice, conoscendone la password e eseguendo il client in nome suo, mentendo così al server. Questo sistema è stato nel tempo sostituito dall'elaborazione di Kerberos.

Kerberos è un servizio di autenticazione sviluppato al MIT nel 1983 nell'ambito del progetto Athena, per fornire agli studenti un'enorme rete di computer che permetta loro di accedere ai file personali da qualsiasi punto della rete. Il MIT lo distribuisce liberamente, per agevolare la sostituzione dello standard per asserzione che presenta notevoli lacune. Kerberos è un'implementazione reale di un protocollo fondato sulla crittografia simmetrica. Forse anche a questo è legata la sua complessità: i meccanismi a chiave pubblica permetterebbero di snellire vari passaggi. Resta il fatto che, proprio grazie all'uso della crittografia simmetrica, Kerberos resisterebbe ad un'eventuale, per quanto improbabile, scoperta di un metodo di fattorizzazione di complessità polinomiale. Il suo scopo è fornire elevati livelli di autenticazione e sicurezza nella

distribuzione di una chiave tra utenti di una rete.
Se, come accade di solito nell'ambito della crittografia privata, due utenti usano per un lungo periodo di tempo la stessa chiave, la sicurezza può essere compromessa. Perciò Kerberos esiste anche in una versione on-line, che permette di produrre una nuova chiave di sessione ogni volta che due utenti vogliono comunicare tra loro. Più precisamente, immaginiamo un ambiente aperto in cui diversi utenti vogliono accedere ai servizi forniti da server distribuiti sulla rete. I server dovrebbere riuscire a:

- limitare l'accesso ai servizi agli utenti autorizzati
- autenticare le richieste di servizi, cioè autenticare gli utenti da cui provengono tali richieste
- evitare, mediante opportuna cifratura, che terzi possano intercettare i contenuti delle comunicazioni

Questi tre obiettivi, di **autorizzazione**, **autenticazione** e **cifratura**, e soprattutto il loro essere tre danno il nome al protocollo. Kerberos infatti era il cane a più teste, normalmente tre, con la coda di serpente, che stava a sorvegliare l'ingresso all'Ade.
Kerberos è legato al numero tre anche per altri motivi. Coinvolge infatti, oltre all'utente, tre server e, nell'idea originale, prevedeva tre componenti per controllare gli ingressi degli utenti nella rete ma solo la prima, quella per l'autenticazione, ha trovato un'implementazione pratica.
Esistono almeno tre possibili attacchi a questo sistema:

- Eva (che può essere un utente della rete) potrebbe accedere ad una postazione (workstation) e impersonare un altro utente
- Eva potrebbe modificare l'indirizzo di rete della propria workstation in modo che la richiesta inviata sembri provenire da un'altra workstation
- Eva potrebbe intercettare gli scambi di messaggi ed organizzare un attacco di tipo replay per ottenere l'accesso a un server, o disturbare il funzionamento della rete

In questi casi Eva può riuscire ad ottenere l'accesso a servizi e dati riservati. Kerberos fornisce un servizio di autenticazione centralizzato, la cui funzione è autenticare gli utenti per i server locali e, volendo, anche i server per gli utenti.
Esistono due versioni di Kerberos, attualmente utilizzate: Kerberos 4, principalmente sviluppata da Steve Miller e Clifford Neuman, e Kerberos 5, progettata da John Kohl e Clifford Neuman.

La seconda versione è stata anche proposta come standard per Internet. Prima di entrare nei dettagli della crittografia impiegata da questo protocollo vediamo meglio le motivazioni che hanno portato alla sua introduzione

1.2 Motivazioni per l'introduzione di Kerberos

Al giorno d'oggi un insieme di utenti lavora (simultaneamente) su un'architettura distribuita costituita da *workstation* (PC) o postazioni dedicate (client) e server distribuiti o centralizzati. Si possono prevedere tre approcci alla sicurezza:

- verificare l'identità degli utenti mediante le singole workstation e affidare **ad ogni server** il compito di autenticare gli utenti.
- chiedere che i sistemi client si autentichino al server, affidando la verifica dell'identitá degli utenti **al sistema client**.
- chiedere all'utente di provare la sua identità per ogni servizio impiegato. E saranno **i server a garantire la propria identità ai client**.

Se ogni server dovesse essere in grado di verificare l'identità degli utenti che si presentano attraverso le strutture client, il carico necessario in termini di memoria e capacità di elaborazione, sarebbe molto grande. Per questo è introdotto nel meccanismo di Kerberos un *server di autenticazione* (AS) che conosce le password di tutti gli utenti e le conserva in un database centralizzato, e che condivide una chiave segreta univoca con ciascun altro server. Tali chiavi dovranno essere distribuite in modo sicuro. Kerberos presuppone una architettura distribuita client/server e utilizza inoltre uno o più server, detti *server Kerberos*, per fornire il servizio di autenticazione. L'impiego di un servizio di autenticazione esterno fidato è fondamentale: server e client confidano entrambi in Kerberos per effettuare la reciproca autenticazione. Ipotizzando che il sistema Kerberos sia ben realizzato, e che il server Kerberos sia sicuro, allora il servizio di autenticazione può essere considerato sicuro.

1.3 Kerberos per tutti

Vediamo in maniera semplice e schematica, comprensibile a tutti, i vari passaggi che permettono a Kerberos di realizzare gli scopi prefissati. Diamo ad ogni soggetto coinvolto nel processo un nome familiare: in questa schema sono indicati questi nomi, abbinati ai nomi più

tecnici usati nella spiegazione più dettagliata che seguirà e alle loro funzioni. Ipotizziamo uno scenario semplificato in cui Cliff, il client, vuole prendere un treno che si chiama Serge. Serge è un treno riservato ai lavoratori dell'impresa MOL.

1. Client - Cliff - richiedente il servizio - viaggiatore, lavoratore alla MOL, che vuole prendere il treno Serge;
2. AS - Trent - autorità fidata che conserva tutte le password abbinate ai nomi degli utenti in un database centralizzato - poliziotto ferroviario che ha un elenco di tutte le persone autorizzate a viaggiare con Serge e le loro firme identificative;
3. TGT - Grant - il ticket-granting server che distribuisce i tickets per i vari altri server - biglietteria centrale della stazione;
4. V - Serge - il server che puó fornire il servizio - treno riservato ai dipendenti;

Cliff ha bisogno di un biglietto per viaggiare a bordo di Serge.
Grant può vendere biglietti per Serge solo a coloro che sono autorizzati ma non si fida sulla parola dell'identità di Cliff. Cliff deve dunque riuscire a dimostrare a Grant la sua identità. Per farlo si reca da Trent. Fá cercare nello schedario della polizia ferroviaria il proprio nome, e mostra la propria firma, spiegando che vorrebbe comperare da Grant, il bigliettaio, un biglietto per viaggiare a bordo di Serge.
Se Trent trova la scheda di Cliff legge il suo nome, confronta la firma fatta da Cliff sul momento con quella conservata nello schedario, e, se c'è coincidenza, gli dà un biglietto su cui scrive l'orario, il nome di Cliff, copia la sua firma dalla scheda e mette il nome del bigliettaio da cui Cliff intende comperare il biglietto: Grant. Per Cliff tale biglietto è illeggibile in quanto scritto in un alfabeto a lui del tutto sconosciuto.
Cliff allora, munito del biglietto ricevuto da Trent, va da Grant, il bigliettaio, e glielo consegna. Si presenta con il suo nome, gli mostra la sua firma, e gli chiede un biglietto per viaggiare a bordo di Serge.
Grant conosce l'alfabeto segreto usato da Trent, e riesce dunque a leggere il biglietto. Se il nome dichiarato da Cliff coincide con quello scritto da Trent sul biglietto, se la firma di Cliff è uguale a quella copiata da Trent sul biglietto, e se il biglietto è stato emesso da poco, allora conclude che Cliff è davvero Cliff. Grant allora controlla nel suo schedario se Cliff è autorizzato a viaggiare a bordo di Serge e gli vende un secondo biglietto. Questo secondo biglietto è scritto

in un alfabeto ancora una volta incomprensibile a Cliff, ma diverso dal primo, e contiene informazioni su chi è Cliff, qual è la sua firma, e il fatto che è autorizzato a viaggiare a bordo di Serge.

A bordo del treno il controllore (che è parte di Serge) chiede a Cliff chi lui sia e gli intima di presentare il biglietto. Cliff allora, ancora una volta, dichiara la propria identità, mostra la propria firma e presenta al controllore il biglietto di cui è in possesso. Il controllore conosce l'alfabeto segreto di Grant e confronta il contenuto del biglietto con le credenziali fornite da Cliff. Se corrispondono augura a Cliff un buon viaggio a bordo di Serge, se non corrispondono Cliff dovrà scendere dal treno, presumibilmente pagando una multa salata.

Alla fine del viaggio, Cliff avrà dimostrato la sua identità e usufruito del servizio. E probabilmente avrà deciso di non usufruirne più, data l'ingiustificata diffidenza che tutti hanno dimostrato nei suoi confronti.

Più in dettaglio, il primo biglietto, agli occhi di chi sa leggerlo, suonerà così:

Io sono Cliff. La mia firma è *Cliff*. Sto andando da Grant. Sono le 8:50. Questo biglietto è valido per 2 ore

Cliff, quando lo presenta a Grant, dirà: io sono Cliff, la mia firma è *Cliff*. Se nome e firma coincidono, se il biglietto è valido, e se Grant riconosce il proprio nome (segno che ha capito quale alfabeto ha usato Trent), dà a Cliff il secondo biglietto che reciterà più o meno così:

Io sono Cliff. La mia firma è *Cliff*. Voglio viaggiare a bordo di Serge. Sono le 9:10. Questo biglietto è valido per 3 ore

Il controllore di Serge ripete l'operazione che ha fatto Grant, come abbiamo visto.

Capitolo 2

Kerberos - Versione 4

Analizziamo più nel dettaglio i vari particolari del complesso meccanismo di Kerberos cercando di capire la loro funzione. Partirò dalla descrizione di una struttura elementare per evolverla verso maggior sicurezza e precisione, mano a mano che si porranno problemi più sottili in termini di sicurezza o efficienza. La sicurezza di Kerberos dipende anche da ipotesi quali presupporre che le workstation siano sicure, cioè che non esista modo per Eva di posizionarsi tra utente e client per catturare la password. Consideriamo inoltre che agli occhi del server di autenticazione, AS, non sussista differenza tra client e server, che vengono visti in modo identico. Infatti sono chiamati entrambi *principals*.

Tutto comincia da un utente, che si connette ad una workstation, accedendo ad un servizio client, e chiede l'accesso al servizio fornito dal server V. Il client chiede la password all'utente e manda al server di autenticazione AS il proprio codice identificativo Id_C, la propria password, P_C, e il codice identificativo del server V al servizio del quale vuole accedere, Id_V.

$$[Id_C|P_C|Id_V]$$

Dove il simbolo | indica la concatenazione.

AS verifica che sia nel proprio database. Controlla che l'utente abbia fornito la password corretta, e se è autorizzato ad accedere al server V. Se la password è corretta AS dovrà convincere ora il server V che l'utente è autentico. Per farlo crea un ticket:

$$Ticket = e_{K_{AS,V}}[Id_C|AD_C|Id_V]$$

dove AD_C è l'indirizzo di rete della workstation da cui si è connesso l'utente, e Id_V è il codice identificativo del server V. AS manda $Ticket$ al client.

Il ticket è cifrato con la chiave $K_{AS,V}$ che è una chiave segreta condivisa tra server di autenticazione AS e server V. Nè il client nè alcun altro estraneo possono leggere il contenuto del Ticket, e tanto meno modificarlo, perchè non conoscono questa chiave.

Il client C ora possiede questo Ticket e può chiedere il servizio a V. Invia al server V:

$$[Id_C|Ticket]$$

V decifra il ticket, servendosi della chiave $K_{AS,V}$ che condivide con AS. Ricava così Id_C e può confrontarlo con Id_C fornito dal client fuori dal ticket. Se i due sono uguali, V concede il servizio.

2.1 Alcune osservazioni

Abbiamo visto che crittografare il ticket assicura che nessuno, nemmeno lo stesso client, possa alterare il suo contenuto. Nel ticket é contenuto Id_V; questa informazione permette al server V di avere una conferma di aver decifrato in maniera corretta il ticket. La presenza invece, sempre nel ticket, di Id_C, oltre ad essere il cuore di questo meccanismo di autenticazione, assicura che l'emissione stessa del ticket avviene proprio per conto del client. Rimane da spiegare la presenza di AD_C.
Per capire la sua funzione consideriamo che Eva riesca ad intercettare il ticket e che, usando Id_C, mandi $[Id_C|Ticket]$ al server V da una workstation che non é quella di indirizzo AD_C. V considererebbe il ticket valido e consentirebbe l'accesso al servizio a chi sta su quest'altra workstation, cioè Eva. Grazie alla presenza di AD_C, invece, il ticket viene considerato valido solo se mandato dalla stessa workstation che ne ha richiesto l'emissione ad AS (sempre che Eva però non riesca temporaneamente a modificare l'indirizzo della sua workstation).

2.2 Primi problemi

Questo primo modello di architettura di autenticazione presenta vari aspetti da migliorare. Cominciamo prendendone in considerazione due:

- sarebbe interessante riuscire a ridurre al minimo il numero di richieste della password all'utente. Potremmo, ad esempio, cercare un metodo che renda il ticket riutilizzabile: ad

ogni sessione di login la workstation può memorizzare il ticket e usarlo poi ogni volta che l'utente vorrà riaccedere a quel servizio. Così facendo però l'utente necessiterà comunque di un nuovo ticket per ogni diverso servizio di cui avrà bisogno e dunque, per ogni servizio, dovrà introdurre nuovamente la password.

- un ben più grave problema di sicurezza: al primo passo della creazione del ticket, viene trasmessa in chiaro la password P_C (problema comune a diversi protocolli di accesso creati negli anni '80). Tale password è dunque soggetta a possibile intercettazioni da parte di Eva.

Per ovviare a questi due primi grandi inconvenienti, nel sistema Kerberos è introdotta una nuova figura: un TGS, un ticket-granting server.

Andiamo dunque a correggere la struttura prima presentata. Divideremo la spiegazione in tre parti che corrispondono ai tre passaggi fondamentali del meccanismo.

2.3 Scambio per l'autenticazione del servizio: il Ticket di assegnamento ticket

Ad ogni sessione di login si verifica il seguente scambio di messaggi:

1. $$C \longrightarrow AS \qquad [Id_C|Id_{TGS}]$$

2. $$AS \longrightarrow C \qquad e_{K_C}[Ticket_{TGS}]$$

$$dove \quad Ticket_{TGS} = e_{K_{TGS}}[Id_C|AD_C|Id_{TGS}|TS_1|Lifetime_1]$$

Spieghiamo a parole questi passaggi schematici:

1. il client C chiede ad AS un ticket di assegnamento ticket ($Ticket_{TGS}$) per conto dell'utente. Manda quindi ad AS il proprio codice identificativo (Id_C) e quello del ticket-granting server di cui intende servirsi: TGS (Id_{TGS}).

2. AS manda al client un ticket crittografato attraverso una chiave generata a partire dalla password dell'utente: K_C. Il client chiede quindi la password all'utente, genera la chiave ed è in grado di estrarre $Ticket_{TGS}$.

Solo l'utente conosce la password, e quindi solo il client C può recuperare il ticket. In questo modo l'utente ottiene il ticket senza trasmettere in chiaro la sua password. Così il secondo problema individuato nella sezione precedente è risolto. Per correggere anche il primo abbiamo detto che vorremmo rendere il ticket riutilizzabile. Immaginiamo però questa situazione: Eva riesce a recuperare un ticket; aspetta che l'utente legittimo si disconnetta dalla sua workstation e poi si siede a quella workstation o modifica l'indirizzo della propria workstation. In entrambi i casi Eva può utilizzare il ticket per ingannare TGS.
È per questo che nel $Ticket_{TGS}$ sono presenti anche un *Timestamp* (chiamato TS_1), che indica il momento dell'emissione del ticket, ed un *Lifetime* (chiamato $Lifetime_1$), che indica la durata di validità del ticket. Il client così possiede un ticket riutilizzabile solo per un periodo di tempo stabilito e dunque, in quel tempo, non deve chiedere la password all'utente per ogni servizio a cui vuole accedere.

Inoltre $Ticket_{TGS}$ è crittografato con chiave segreta nota solo ad AS e TGS. Il suo contenuto non è dunque modificabile. Viene poi crittografato una seconda volta con una chiave generata direttamente dalla password dell'utente. In questo modo $Ticket_{TGS}$ è recuperabile solo dall'utente corretto, che comunque rimane **non** in grado di leggerlo, e dunque il sistema di autenticazione è ancora valido.

2.4 Scambio per ottenere il Ticket di servizio

Ora il client possiede $Ticket_{TGS}$, il ticket di assegnamento ticket, e si accinge a richiedere un secondo ticket, che possiamo chiamare ticket di servizio.

1.
$$C \longrightarrow TGS \qquad [Id_C|Id_V|Ticket_{TGS}]$$

2.
$$TGS \longrightarrow C \;\; [Ticket_V]$$

$$dove \qquad Ticket_V = e_{K_V}[Id_C|AD_C|Id_V|TS_2|Lifetime_2]$$

Spieghiamo meglio anche questi due passaggi:

1. il client, per conto dell'utente, chiede un ticket di servizio a TGS. Per farlo invia a TGS, insieme a $Ticket_{TGS}$, il proprio identificativo (Id_C) e quello del server V (Id_V) al servizio del quale vuole accedere.

2. TGS decrittografa $Ticket_{TGS}$. Lo può fare perchè è stato cifrato attraverso la chiave K_{TGS}, condivisa tra AS e TGS. A seguire poi verifica di aver decifrato correttamente controllando Id_{TGS}, verifica che il $Ticket_{TGS}$ non sia scaduto, confronta Id_C e AD_C che ricava dalla decifrazione, con le informazioni che il client gli ha mandato in chiaro. Se coincidono, il client, e dunque l'utente, viene autenticato. TGS controlla che a questo utente sia garantito l'accesso al servizio di V e, in caso affermativo, emette $Ticket_V$.

Siccome sia TGS che AS sono server, funzionano in modo simile, e dunque i ticket hanno, nella struttura base, la stessa forma. Se l'utente vorrà accedere al servizio del server V in un secondo tempo, il client potrà riutilizzare il $Ticket_V$ di prima senza richiedere la password. $Ticket_V$ è crittografato con una chiave K_V, nota solamente a TGS e V. Esso non é dunque modificabile da nessuno, nemmeno da C, esattamente come $Ticket_{TGS}$.

2.5 Accesso al servizio

$$C \longrightarrow V \qquad [Id_C, Ticket_V]$$

Il client, sempre per conto dell'utente, chiede l'accesso al servizio mandando al server V il proprio identificativo (Id_C) e il $Ticket_V$. V decifra il ticket, usando la chiave K_V che condivide con TGS, e confronta i due identificativi del client; se coincidono concede il servizio.

2.6 Ulteriori problemi

Nonostante i miglioramenti apportati attraverso l'introduzione di TGS, esistono ancora importanti aspetti riguardo ai quali è bene interrogarsi. Il primo riguarda la durata della validità dei tickets: se un ticket ha validità troppo breve, l'utente sarà costretto a reintrodurre la password mentre, se è troppo lunga, Eva potrà inserirsi nel meccanismo con un attacco a replay intercettando $Ticket_{TGS}$, aspettando la disconnessione dell'utente legittimo dalla workstation e sostituendoglisi. È dunque importante riuscire a dimostare che chi usa un ticket sia colui per cui il ticket è stato emesso. Altra questione riguarda la possibilità che, così come l'utente si autentica presso il server, sia previsto che sia anche il server ad autenticarsi nei confronti

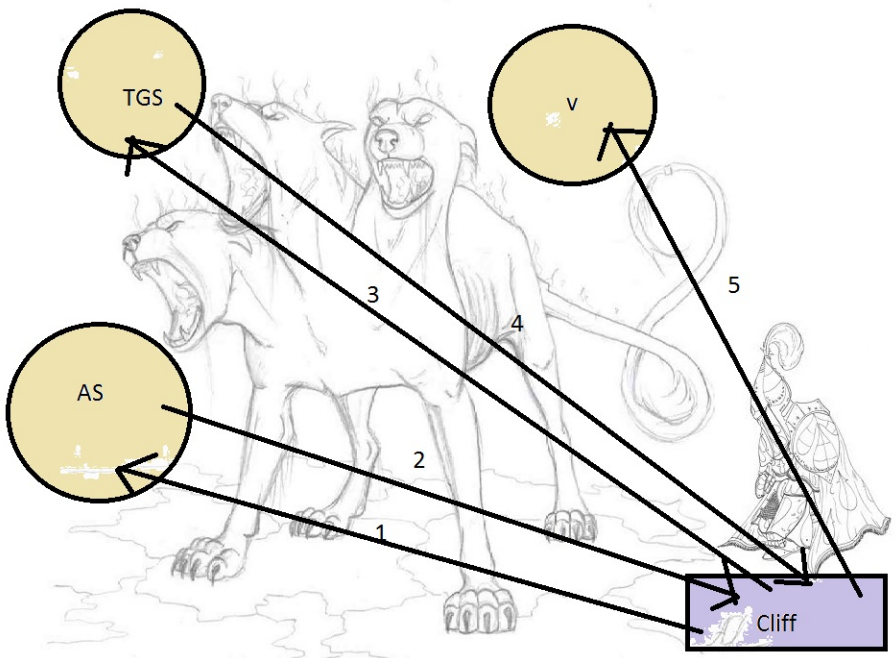

Figura 2.1: Schema base di Kerberos

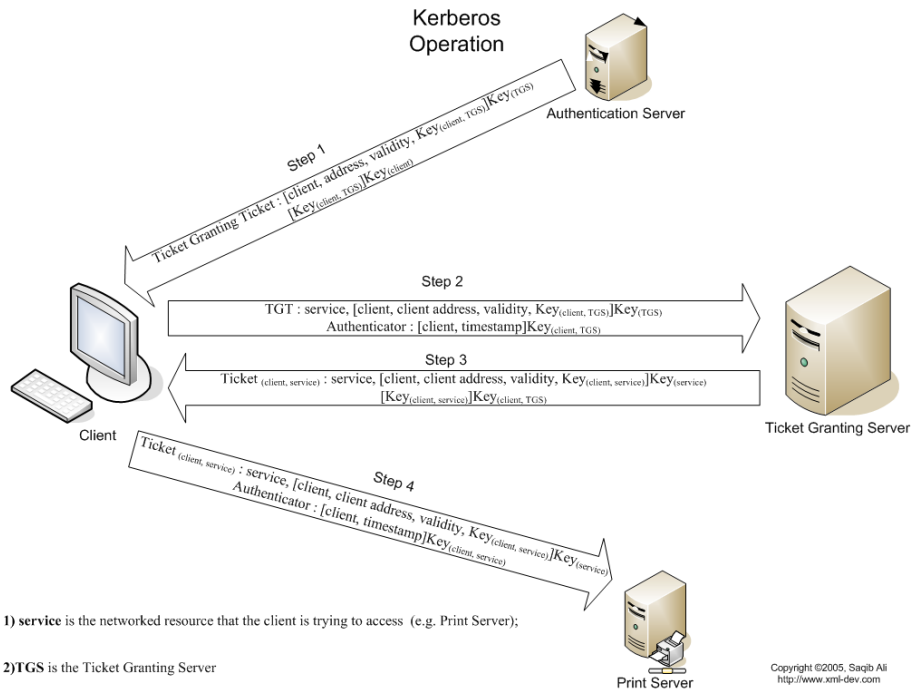

Figura 2.2: Schema base più dettagliato di Kerberos

dell'utente. Eva potrebbe altrimenti configurare un falso server ed impadronirsi cosí delle informazioni riservate del client.

2.7 Introduzione delle chiavi di sessione

Se temiamo che Eva intercetti il $Ticket_{TGS}$ e lo utilizzi, prima della scadenza, spacciandosi per un altro utente, possiamo introdurre una *chiave di sessione* che impedisca questo tipo di attacco. Una chiave di sessione è una chiave crittografica, quindi un'informazione segreta, fornita da AS, in modo sicuro, al client e a TGS. Il client e TGS possono dimostrare reciprocamente la propria identità mostrando all'altro di conoscere questa informazione. Ripercorriamo allora tutto il sistema con le modifiche:

1.
$$C \longrightarrow AS \qquad [Id_C|Id_{TGS}|TS_1]$$

2.
$$AS \longrightarrow C \qquad e_{K_C}[K_{C,TGS}|Id_{TGS}|TS_2|Lifetime_2|Ticket_{TGS}]$$

$$dove \quad Ticket_{TGS} = e_{K_{TGS}}[K_{C,TGS}|Id_C|AD_C|Id_{TGS}|TS_2|Lifetime_2]$$

$K_{C,TGS}$ è detta *chiave di sessione* e viene condivisa dal client e de TGS, ma è generata da AS. Il client la ottiene perchè gli viene inviata da AS e TGS la legge da $Ticket_{TGS}$, che lui riesce a decifrare.
Possiamo osservare che AS manda al client anche altre informazioni come Id_{TGS},TS_2 e $Lifetime_2$ in modo che questi possa ricordarsi, in qualsiasi momento, che il ticket di cui dispone è per il server TGS e possa controllare la sua scadenza. A questo punto il client dispone di

- $Ticket_{TGS}$
- $K_{C,TGS}$ chiave di sessione, condivisa con TGS

Può dunque contattare TGS.

1.
$$C \longrightarrow TGS \qquad [Id_V|Ticket_{TGS}|Autenticatore_C]$$

2.

$$TGS \longrightarrow C \quad e_{K_{C,TGS}}[K_{C,V}|Id_V|TS_4|Ticket_V]$$

$$dove \quad Autenticatore_C = e_{K_{C,TGS}}[Id_C, AD_C, TS_3]$$

$$e \quad Ticket_V = e_{K_V}[K_{C,V}|Id_C|AD_C|Id_V|TS_4|Lifetime_4]$$

L'autenticatore viene usato una sola volta ed è di breve durata, piccoli accorgimenti per scongiurare gli attacchi a replay. TGS decrittografa $Ticket_{TGS}$ usando K_{TGS} (condivisa sempre tra AS e TGS).
TGS usa poi la chiave di sessione $K_{C,TGS}$ per decrittografare l'$Autenticatore_C$. TGS ottiene dunque Id_C e AD_C sia dall'autenticatore che dal ticket e può confrontarli.Se sono uguali considera verificata l'identità del client.
Viene istituita da TGS una chiave di sessione anche tra il client ed il server V:

$$C \longrightarrow V \quad [Ticket_V|Autenticatore_{C_2}]$$

$$dove \quad Autenticatore_{C_2} = e_{K_{C,V}}[Id_C|AD_C|TS_5]$$

Il server V decrittografa $Ticket_V$ usando la chiave K_V, condivisa tra TGS e V. Recupera così $K_{C,V}$ e può decifrare anche l'autenticatore. Verifica così l'identità del client.

A questo schema, ormai completo, possiamo aggiungere un ultimo passaggio, facoltativo, che permette al client di avere la certezza di comunicare con il vero server V. È infatti sufficiente che V invii al client:

$$V \longrightarrow C \quad e_{K_{C,V}}[TS_5 + 1]$$

Il server rimanda dunque al client il valore del Timestamp ottenuto dall'autenticatore del client aumentato di 1, e crittografato con la chiave di sessione $K_{C,V}$. L'utilizzo proprio di quella chiave di sessione assicura al client che solo il server V può aver generato quel messaggio, e decifrando il messaggio il client ottiene il Timestamp dell'autenticatore e sa che non è il replay di una vecchia risposta.
Grazie a questo metodo, inoltre, C e V condividono una chiave di sessione $K_{C,V}$ che possono usare per scambiarsi altri messaggi o per trasmettersi una nuova chiave temporanea attraverso un canale sicuro. Se $K_{C,V}$ venisse intercettata, sarebbe compromessa tutta la comunicazione tra C e V, da quel momento in poi.

2.8 Realm differenti

L'ambiente Kerberos è quindi composto da tanti client, da un server Kerberos, che conserva in un database tutti i codici degli utenti registrati, abbinati alle loro passwords, e da altri server (V) fornitori di servizi, ognuno dei quali condivide con il server Kerberos una chiave segreta. Questa struttura base prende il nome di *realm*. Client e server che agiscono nella stessa rete costituiscono un realm. È tuttavia possibile che un client voglia accedere ad un servizio fornito da un server di un altro realm. Kerberos prevede questa possibilità che consiste, fondamentalmente, nella autenticazione tra due realm. Questo meccanismo si fonda sulla condivisione di una chiave segreta tra i server Kerberos dei due realm differenti. Ogni server Kerberos ha la responsabilità di autenticare i propri utenti e la struttura si regge sulla fiducia che ogni server Kerberos ha verso il suo corrispondente nel secondo realm. Supponiamo che un utente voglia accedere al servizio fornito da un server W di un realm che non è il suo: dovrà dunque procurarsi un ticket per W.
L'utente, attraverso la struttura client, richiede al suo server TGS un ticket di assegnamento ticket per il TGS dell'altro realm, detto TGS remoto. Chiede poi a questo secondo TGS un ticket per il server W. Su tale ticket verrà indicato che l'utente è autenticato su un altro realm. Poi W deciderà se concedere o meno il servizio richiesto. Se i realm sono molti il sistema non sarà efficiente: richiede infatti un numero elevato di chiavi segrete per permettere la comunicazione sicura tra server Kerberos dei vari realm (per N realm servono $\frac{N(N-1)}{2}$ chiavi segrete).

Capitolo 3

Kerberos - Versione 5

La versione 5 di Kerberos (K5), pur mantenendo la forma base della versione 4, apporta modifiche volte a risolvere problemi e vulnerabilità del sistema, di natura ambientale e tecnica. Proverò a fornire una panoramica di questi cambiamenti identificando le debolezze nella versione 4 e osservando come vengono eliminate nella 5. Molte migliorie vengono apportate mediante l'utilizzo di flags dei ticket. Il campo FLAG è un campo del ticket che può supportare funzionalità differenti a seconda della versione di Kerberos considerata.

3.1 A livello di crittografia

- Kerberos 4 impiega DES. DES non è un sistema adattabile a diverse piattaforme e non è un sistema a sicurezza assoluta. Nella versione 5 il tipo di crittografia può variare, anche nel numero e nel tipo delle chiavi, informazioni specificate su un identificatore che accompagna il testo cifrato.
- I ticket forniti ai client vengono crittografati due volte: la prima volta con la chiave segreta del server di destinazione in modo da garantire l'integrità del ticket, e la seconda con la chiave del client. In questo modo il client può accedere a informazioni quali la scadenza del ticket. Il secondo passaggio non è indispensabile. È uno spreco di lavoro a livello computazionale, quindi Kerberos 5 lo abolisce.
- Kerberos 4 usa DES in una variante alternativa detta PCBC (Propagating Cipher Block Chaining). È una tecnica vulnerabile ad attacchi con scambi di blocchi cifrati. PCBC include il controllo di integrità nel meccanismo. In K5 tale controllo è separato dall'operazione crittografica vera e propria, che sfrutta DES in modalità standard CBC.

3.2 Durata dei ticket

In Kerberos 4 la durata massima del ticket è:

$$2^8 * 5 = 1280 minuti \approx 21 ore$$

che sono poche per alcune applicazioni. In Kerberos 5, i ticket includono:

1. *from*: tempo di inizio della validità del ticket;
2. *till*: tempo di fine della validità del ticket;
3. *rtime*: tempo di rinnovo richiesto;

La durata del ticket è dunque arbitraria. Se un ticket ha durata lunga può essere intercettato e sfruttato per un periodo di tempo importante. Se la durata è breve, si dovrà considerare il sovraccarico dovuto all'acquisizione di nuovi ticket. Per gestire queste problematiche viene anche introdotto nei ticket il flag RENEWABLE. Un ticket con questo flag attivo è *rinnovabile*, cioè include due date di scadenza: una vera e propria e una che rappresenta l'ultimo valore consentito come data di scadenza. Il rinnovo del ticket è ad opera, naturalmente, del client che può richiedere una nuova data di scadenza a TGS. Se la nuova scadenza è entro i limiti dell'ultimo valore consentito, TGS può emettere un nuovo ticket con una nuova scadenza. TGS può così rifiutarsi di rinnovare un ticket rubato, ad esempio.
Il client può anche chiedere ad AS un ticket con flag MAY-POSTDATE. Esso dà accesso all'emissione, da parte di TGS, di un ticket con flags POSTDATED e INVALID. In seguito il client potrà inoltrare il ticket postdatandolo per la convalida. È un meccanismo utile per snellire operazioni lunghe su server che richiedono periodicamente un ticket; infatti il client può ottenere in anticipo un certo numero di ticket di sessione con valori temporali ben distanziati. Inizialmente è valido solo il primo. Quando l'esecuzione lo richiede il client può convalidare il nuovo ticket. In questo modo il client non dovrà usare molte volte il suo ticket di assegnamento ticket.

3.3 Server che dialoga con un altro server

Nella versione 5 di Kerberos un server può richiedere un servizio ad un altro server per conto del client. Per farlo il client chiede ad AS un ticket di assegnamento ticket con flag PROXIABLE. Quando TGS riceve $Ticket_{TGS}$, può emettere un ticket di servizio con indirizzo

di rete modificato. Il server che riceverà tale ticket può decidere se fidarsi o chiedere altre informazioni per accertarsi della validità del ticket e dell'identità del mittente.

3.4 Dialogo tra realm differenti: la procedura di *forwarding*

Operare tra vari realm nella versione 4 prevede la presenza e sicurezza di un numero di chiavi segrete molto alto. La versione 5 prevede meno relazioni. Innanzi tutto nel $Ticket_{TGS}$ viene aggiunto un campo realm che identifica il realm dell'utente. Esiste inoltre un flag che consente una variante, utile perché un client riesca ad ottenere un servizio da un server di un realm differente da quello a cui appartiene. Se un ticket, emesso da AS, ha il flag FORWARDABLE impostato, il server TGS puó emettere al richiedente un ticket di assegnamento ticket con un indirizzo di rete diverso e il flag FORWARDED. Questo nuovo ticket può essere presentato a un server TGS remoto, ossia di un altro realm.
Questa procedura, detta *forwarding*, permette ad un client di ottenere un servizio da un server di un altro realm, senza che però il server Kerberos del proprio realm debba condividere una chiave segreta col server Kerberos remoto. Un client può risalire l'albero, arrivare ad un nodo comune, e ridiscendere giungendo nel realm del server remoto. Ad ogni movimento il client necessita di un ticket di assegnamento ticket da presentare al TGS successivo nel percorso.

3.5 Combattere gli attacchi alle password: processi di *preautenticazione*

Eva, in Kerberos 4, può organizzare un attacco alle password. AS, infatti, invia al client un messaggio cifrato con una chiave segreta ricavata dalla password dell'utente. Eva può intercettare il messaggio e forzarlo provando varie password. Se ha successo per un tentativo, a quel punto conosce la password del client e potrà utilizzarla per accedere al materiale segreto protetto da Kerberos. Non è così improbabile che Eva riesca ad indovinare la password dell'utente: le password umane sono spesso facili da individuare con attacchi come, ad esempio, gli *attacchi del dizionario*. Esistono inoltre molti metodi, come quello del *cavallo di Troia* (Trojan Horse) che creano una falsa finestra di login per catturare le password (operazione di *phishing*). Eva potrebbe anche cercare di accedere al database di AS direttamente, dove sono memorizzate tutte le password. Per questo esse vengono conservate criptate dal server Kerberos. Kerberos 5 prevede un sistema, detto *preautenticazione*, che, per lo meno, complica, pur non impedendo, gli attacchi alle password. Nel ticket la preautenticazione compare con il

flag PREAUTHENT. Il server AS che ha ricevuto la richiesta del client ha autenticato il client prima di emettere il ticket. Vediamo un po' più nel dettaglio questa preautenticazione in due possibili implementazioni:

- Se un utente vuole ottenere un ticket, deve inviare al server AS un blocco di preautenticazione che contenga:

  $$C \longrightarrow AS$$

  1. un valore casuale;
  2. numero della versione di Kerberos utilizzata;
  3. codice timestamp;

  crittografati attraverso la chiave basata sulla password del client.

  $$AS \longrightarrow C$$

  AS decrittografa il blocco e restituisce il $Ticket_{TGS}$ solo se il timestamp decrittografato sta nella finestra di tempo legittima.

- *implementazione MIT*: implementazione con l'uso di una smart-card. È indicato dal flag HW-AUTHENT. Una smart card genera continuamente nuove passwords che vengono incluse nel messaggi preautenticati. Le passwords sono generate a partire dalla password dell'utente, ma la card deve introdurre delle modifiche casuali. Questo complica, tuttavia non impedendo, gli attacchi alle password.

3.6 Strategie per combattere gli attacchi a replay

Una attacco a replay, o attacco di replica, può essere organizzato da Eva. Può infatti intercettare dei messaggi in uno scambio privato e riproporli successivamente ad Alice o Bob. Saranno crittografati con le chiavi corrette, e possono quindi essere considerati autentici da chi li riceve. Eva dunque può sfruttare questa modalità per ottenere infomazioni riservate o, semplicemente, per confondere il dialogo tra Alice e Bob. Kerberos 5 prevede alcuni piccoli accorgimenti che tentano di scongiurare eventuali attacchi a replay.

3.6.1 Le sottochiavi di sessione

Abbiamo visto che ogni ticket contiene una chiave di sessione, sfruttata dal client per crittografare l'autenticatore. La chiave di sessione può poi essere utilizzata da client e server

per scambiarsi altri messaggi in modo sicuro. Il ticket però è riutilizzabile quindi Eva potrebbe intercettare messaggi in una sessione e riproporli al client o al server in una nuova sessione spacciandoli per autentici. Per bloccare questo tipo di attacco, in K5, client e server, ad ogni nuova connessione, ovvero ad ogni utilizzo del ticket, concordano una nuova *chiave di sottosessione*. In particolare è il client che, ad ogni utilizzo nuovo del ticket, introduce una sottochiave nell'autenticatore. Se il client omette tale passaggio, per quella connessione si userà la chiave di sessione principale.

3.6.2 Nonce

Nel $Ticket_{TGS}$ viene aggiunto un *nonce* scelto dal client ed inviato ad AS. Un nonce è un valore casuale, che cambia ad ogni invio del messaggio, che testimonia che la risposta non è un replay inviato da Eva.

3.6.3 Numero sequenziale

Nello scambio server/client si può aggiungere un numero sequenziale, che specifica il numero iniziale della sequenza. I messaggi verranno numerati in sequenza per rilevare eventuali ripetizioni. Se un'entità riceve un messaggio con un numero di sequenza già usato, lo identifica come una replica. Il problema di questo metodo è che richiede alle entità di tenere traccia di tutti i numeri usati (come anche dei nonce).

Anche il timestamp, di cui ho già discusso, può essere visto come una strategia per combattere questo tipo di attacco. È importante però, affinchè i timestamp siano di qualche utilità, che gli orologi delle varie entità siano sincronizzati tra loro, e questo è un problema difficile.

3.7 Ulteriori piccole modifiche

Kerberos 4 è limitato ad un certo tipo di indirizzi rete. Nella versione 5 viene esteso il campo di indirizzi gestibili, indicandone tipo e lunghezza.
Da ultimo nella versione 5 un ticket può essere impostato con il flag INITIAL. Esso evidenzia che il ticket è stato emesso da AS, e non da TGS. In K4 un ticket INITIAL può essere solamente un ticket di assegnamento ticket. Nella versione più recente invece il client può ottenere anche il ticket di servizio direttamente da AS. Così un server, per esempio un server per il cambio di

password, può voler sapere se la password del client sia stata controllata recentemente.

Dopo aver visto una breve panoramica delle principali limitazioni di Kerberos, entreremo più nel dettaglio nella crittografia utilizzata.

3.8 Principali limitazioni di Kerberos

I principali aspetti di Kerberos che lo rendono ancora un sistema non del tutto sicuro sono:

- non protegge efficacemente contro la possibilità che la password dell'utente sia compromessa.
- richiede un canale sicuro che gestisca la password dell'utente, e la presenza di AS, un server di autenticazione sicuro e fidato che dovrà stare, protetto, in un'area fisicamente sicura.
- il server AS deve essere costantemente in funzione.
- Kerberos, nell'implementazione MIT, non può essere esportato (come DES d'altronde). La sua diffusione fuori dagli USA è legata al fatto che è stato implementato in altri paesi.
- contro l'attacco a cavallo di Troia non c'è soluzione: un utente seduto ad una workstation non può sapere se il server sia stato manomesso.
- se la password di un server o di un utente viene scoperta è possibile poi usarla per decifrare altri tickets e ingannare altri principals. È il meccanismo detto dello *spoofing*.
- i programmi che usano Kerberos devono essere tutti modificati attraverso un'operazione di "Kerberizing" (o Cerberizzazione). Può essere un lavoro complesso.

A livello storico, nella prima implementazione, le password di autenticazione viaggiavano in chiaro e quindi potevano essere facilmente intercettate da Eva, compromettendo l'intero processo.

Capitolo 4

Tecniche di crittografia in Kerberos

Kerberos 4 utilizza DES. In particolare ogni utente condivide con AS una chiave K_c DES segreta. DES è un sistema crittografico introdotto nel 1974, non è un sistema a sicurezza assoluta. È un cifrario a blocchi e lo spazio dei testi in chiaro, M, coincide con lo spazio dei testi cifrati, C, e, in entrambi i casi, si tratta di blocchi di 64 bit:

$$M = C = \{0,1\}^{64}$$

Lo spazio delle chiavi invece, K, è:

$$K = \{0,1\}^{56}$$

DES si fonda su una rete di permutazioni (lineari) e sostituzioni (non lineari). In particolare esiste un blocco di operazioni che viene ripetuto 16 volte impiegando una chiave sempre diversa, detta *chiave di round*.

Lo schema base dell'architettura di DES è descritto in figura.

IL messaggio iniziale, di 64 bit, attraversa una permutazione iniziale il cui output, ancora di 64 bit, viene visto come due blocchi da 32 bit che prendono il nome di L_0 (L di *left*) ed R_0 (R di *right*). Ad ogni passaggio avviene:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

dove $\oplus$ prende il nome di operazione XOR (è la somma vettoriale in $\mathbb{Z}_2^{32}$). Le chiavi, $k_1, k_2, \ldots k_{16}$ sono chiavi di 48 bit ricavate dalla chiave iniziale da un'operazione di scorrimento laterale verso sinistra.

Analizziamo ora più approfonditamente il funzionamento della funzione f. Essa prende come

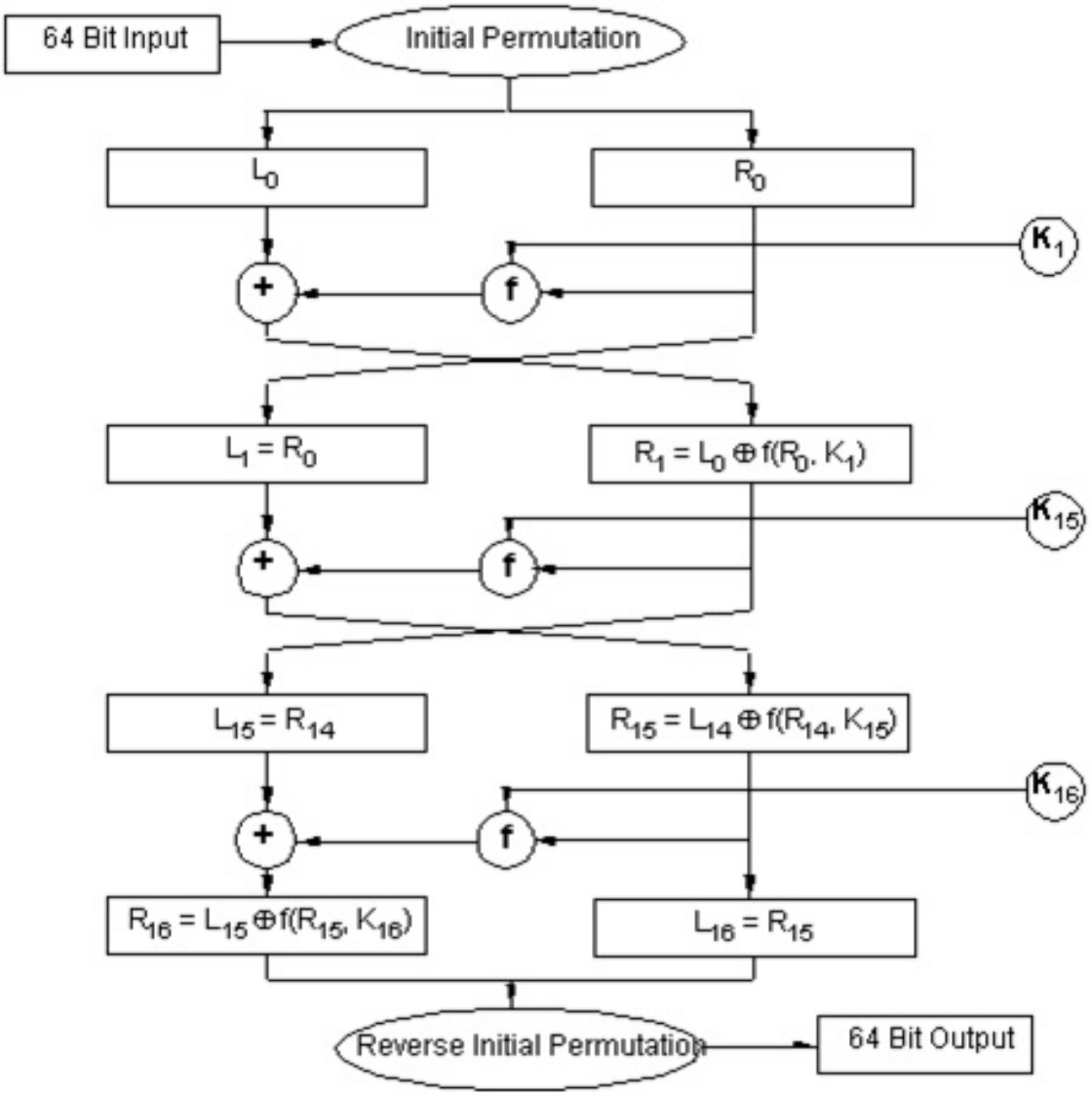

Figura 4.1: Schema base di DES

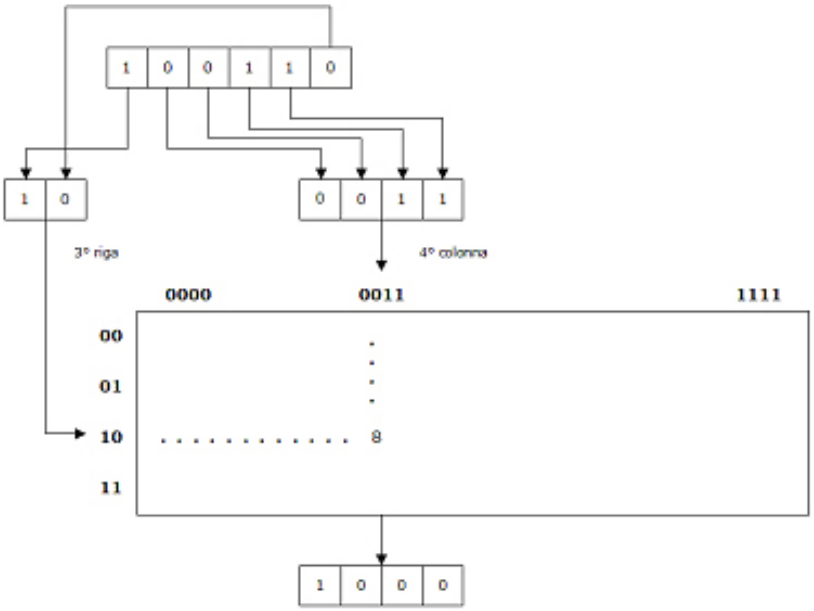

Figura 4.2: Esempio di funzionamento di una S-box

input R_{i-1} (di 32 bit) e k_i (di 48 bit). Il primo passaggio è dunque un'espansione E di R_{i-1} al blocco $E(R_{i-1})$, di 48 bit. L'operazione XOR poi somma $E(R_{i-1})$ e k_i, restituendo un altro blocco di 48 ($= 8 \times 6$) bit.

Abbiamo dunque 8 blocchi da 6 bit. Ognuno di questi blocchi passa attraverso una S-box, S_i, di DES, e viene trasformato (e questa è la parte non lineare del meccanismo) in un blocco di 4 bit. In particolare una S-box può essere rappresentata come una matrice 4×16. Dato il blocco di 6 bit, considero primo e ultimo come legati, a indicare il numero di riga, n, della S-box, e i 4 centrali ad indicare il numero di colonna, m, della S-box. Al posto (n, m) della S-box si trova un numero che può essere scritto in binario con 4 bit. Questi 4 bit vengono sostituiti al blocco iniziale.

Ad esempio, sia blocco iniziale (poniamo che sia il primo blocco della stringa): 011001. 01, cioè 1, è il numero della riga, e 1100, cioè 12, è il numero della colonna. Al posto $(1, 12)$ della S_1-box di DES sta il numero 12 dunque a 011001 si sostituisce il blocco 1100, la rappresentazione di 12 in base 2.

Avrò dunque una nuova stringa di bit lunga $8 \times 4 = 32$ bit. Permutando ulteriormente questi 32 bit ottengo un blocco che, sommato (sempre con XOR) a L_{i-1} sarà il mio nuovo R_i.

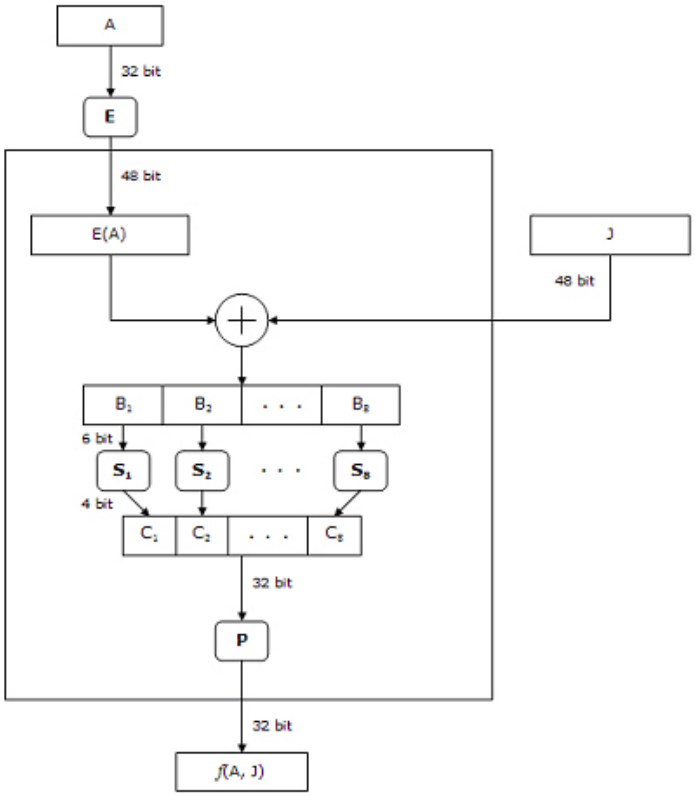

Figura 4.3: Schema di funzionamento della f che agisce ad ogni passaggio di DES

La decrittografia funziona allo stesso modo ma con la sequenza delle chiavi di round invertita: $k_{16}, k_{15}, \ldots, k_1$.

4.1 DES in modalità CBC (Cipher Block Chaining)

È la modalità standard di uso di DES. Permette di cifrare i blocchi $m_1, \ldots, m_n$ di un testo in chiaro in modo concatenato. In questo modo la cifratura di due blocchi uguali produrrà blocchi cifrati differenti. Eva, inoltre, non potrà alterare l'ordine dei blocchi senza alterare il messaggio. Il meccanismo di concatenazione è molto semplice: per prima cosa scegliamo un blocco $c_0 \epsilon \{0,1\}^{64}$. È scelto arbitrariamente ed è detto *vettore di inizializzazione*. Esso può anche essere noto. A questo punto:

$$c_1 = E_k(m_1 \oplus c_0)$$

$$c_2 = E_k(m_2 \oplus c_1)$$

$$\ldots$$

$$c_i = E_k(m_i \oplus c_{i-1})$$

Il meccanismo di decifrazione sarà:

$$D_k(c_1) = m_1 \oplus c_0 \Rightarrow m_1 = D_k(c_1) \oplus c_0$$

$$m_2 = D_k(c_2) \oplus c_1$$

$$\ldots$$

$$m_i = D_k(c_i) \oplus c_{i-1}$$

4.2 Modalità PCBC (Propagating Cipher Block Chaining)

È un'estensione di CBC. Un errore in un blocco di testo cifrato si propaga a tutti i blocchi successivi del messaggio, rendendoli inutilizzabili. Questo sistema permette di raggiungere, in una sola operazione, due obiettivi: integrità del messaggio e sicurezza crittografica.

$$c_1 = E_k(c_0 \oplus m_1 \oplus m_0)$$

$$c_2 = E_k(c_1 \oplus m_2 \oplus m_1)$$

$$\dots$$

$$c_i = E_k(c_{i-1} \oplus m_i \oplus m_{i-1})$$

La decifrazione è analoga:

$$m_1 = c_0 \oplus m_0 \oplus D_k(c_1)$$

$$m_2 = c_1 \oplus m_1 \oplus D_k(c_2)$$

$$\dots$$

$$m_i = c_{i-1} \oplus m_{i-1} \oplus D_k(c_i)$$

4.3 Processo di trasformazione della password in chiave

La password dell'utente è una stringa di caratteri di lunghezza arbitraria. Ogni carattere deve poter essere scritto in formato ASCII su 7 bit. La password viene convertita in una chiave crittografica che poi viene conservata del database di AS. Vediamo come ciò avviene nella versione 4:

1. la stringa di caratteri viene convertita in una stringa di bit semplicemente sostituendo ad ogni carattere i suoi 7 bit corrispondenti. La stringa di bit sarà di lunghezza variabile;
2. dovendo creare una chiave DES devo ridurre il numero bit a 56. Per farlo piego *a ventaglio* la stringa di bit e sommo bit a bit mediante l'operazione XOR i bit che si vengono ad affiancare (essenzialmente questa è un'operazione di hashing);
3. avendo ottenuto una chiave di 56 bit posso usarla per applicare DES: la password originaria viene crittografata in modalità CBC di DES con la chiave ottenuta dalla password stessa (questa operazione di crittografia sulla password non aggiunge alcun livello di protezione supplementare);
4. l'ultimo blocco di 64 bit ottenuto nel processo è la chiave di output associata alla password

Questo meccanismo è analogo al funzionamento di una funzione hash crittografica che mappa una password su un codice hash da 64 bit.
Nella versione 5 la password dell'utente viene concatenata con un SALT: un numero, o stringa di bit, che cambia ad ogni esecuzione del client. Questo serve ad introdurre nella chiave una dipendenza dalla macchina su cui è stata generata. Virtualmente nella K5 può essere usato qualsiasi algoritmo crittografico, ma quelli più utilizzati sono ancora DES e 3-DES.

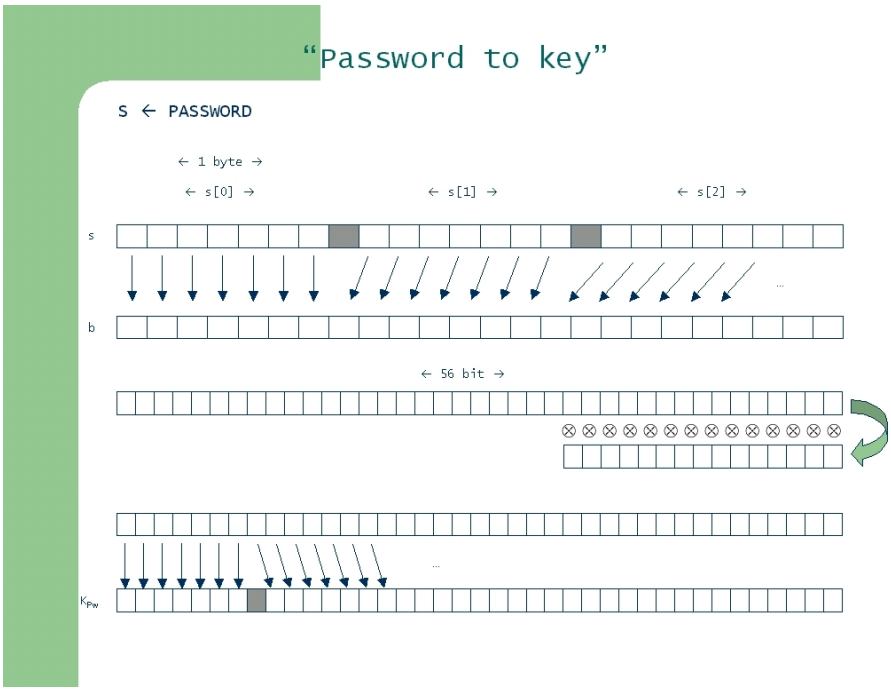

Figura 4.4: Meccanismo di generazione della chiave a partire dalla password

Conclusioni

Il sistema Kerberos, come sistema di autenticazione, ha limitazioni forti. In primo luogo la scarsa flessibilità: non è adattabile a molte piattaforme, se non attraverso rilevanti modifiche sui programmi, chiamate Kerberizing , che comportano un costo molto elevato. Sembrerebbe quindi logico, quasi naturale, che Kerberos, nella pratica, venisse sorpassato , e soppiantato, da implementazioni di sistemi per l'autenticazione che utilizzino, almeno in parte, dei meccanismi a chiave pubblica. Perchè questo non avviene? Possiamo azzardare alcune ipotesi: intanto Kerberos è fruibile gratuitamente ed è già implementato. Quindi, pur nella complessità, la sua disponibilità immediata costituisce un notevole vantaggio. In secondo luogo, Kerberos si regge sulla fiducia che i principals distribuiti nella rete nutrono nei confronti del server centrale. Per garantire l'affidabilità di tale server, se, in un sistema a chiave simmetrica, è sufficiente assicurare fisicamente l'inviolabilità del database, nascondendolo in un luogo segreto e il più possibile inaccessibile, l'uso di chiavi pubbliche prevede la presenza di certificati digitali. Tali documenti vengono generalmente rilasciati da autorità di certificazione americane. Così nel sistema entrerebbero automaticamente in gioco relazioni politiche internazionali complesse e la sicurezza dei dati acquisterebbe una forte dipendenza dai rapporti tra nazioni.

Nell'elaborazione della mia tesi ho trovato stupefacente notare quanto poco, nella ricerca di soluzione ai problemi di autenticazione, influiscano quelle caratteristiche proprie dell'individuo che ci permettono di riconoscere, in un certo senso "autenticare", un amico, un conoscente, nella realtà fisica di ogni giorno. Nell'era di diffusione dei rapporti digitali ci identifica, agli occhi della società vista nella sua accezione più allargata, se non altro a livello numerico, una cifra, un codice, una chiave, una password.

Si riscontra però una notevole somiglianza tra il meccanismo evolutivo di un sistema crittografico, quale può essere Kerberos, analizzato in questa tesi, e quello naturale, "darwiniano", che porta avanti la crescita del sistema Terra. Di fronte al presentarsi di un problema, che in crittografia sarà una falla, una debolezza del sistema, e per i fringuelli di Darwin poteva essere

rompere l'involucro del seme col becco, gli esperti introducono delle modifiche, che sceglieranno se rendere note o meno, testando come queste possano risolvere, almeno in parte, il problema rilevato. Allo stesso modo la natura, con assoluta casualità, fa nascere, da qualche parte nel mondo, un fringuello con il becco più robusto degli altri. è la selezione naturale a stabilire se quel fringuello sia avvantaggiato rispetto ai suoi simili e dunque meriti di sopravvivere e tramandare questo carattere alla sua discendenza, così come in ambito crittografico, non esistono preferenze, raccomandazioni, ma regna la democrazia: se una modifica è utile, valida, sopravvive, fino all'avvento di un sistema che offra maggiori garanzie di sicurezza o maggiore agilità, se è dannosa, o semplicemente inutile, o ancora aumenta la complessità del sistema, viene abolita.

Merita forse una riflessione più profonda sapere che tutto il sistema Kerberos fonda sulla *fiducia* che le entità ripongono nel server di autenticazione. Fiducia, un valore che sembra oggi si stia perdendo nella società umana, nei rapporti concreti, quotidiani, immediati, lo ritroviamo in un ambito, a mio parere così impersonale, distaccato, come quello del digitale, del tecnologico; un livello allo stesso tempo più profondo, in termini tangibili, costruttivi, e più superficiale, in termini umani. Fiducia, così indecifrabile, inspiegabile, non razionale, nell'umano, che viene assunta a fondamento di qualcosa di logico, schematico, efficiente, nel virtuale.

Osservando il funzionamento di Kerberos, ho potuto dunque osservare che anche il mondo digitale, che è essenzialmente artificiale, costruito dall'uomo, dalla mente, imiti e riprenda meccanismi naturali, riproponendoli in analogie, volute o involontarie, più o meno fedeli al dato fisico. Questo apre una breccia nella aura di aridità e impersonalità che circonda spesso il mondo dei computer, soprattutto agli occhi dei profani e, a mio modesto parere, conferisce una nuova luce di umanità al progresso tecnologico. Luce, che potrà continuare a brillare però solo tenendo ben presente quell'imprescindibile distinzione tra utilitaristici rapporti telematici e relazioni umane, meno igieniche, sicure, ma sicuramente più vive.

Bibliografia

Crittografia e sicurezza delle reti, William Stalling, Milano, Ed Mc Graw-Hill, 2004

Cryptography: theory and practice, Stinson Douglas R., CRC, 1995

Aritmetica, crittografia e codici W. M. Baldoni, C. Ciliberto, G. M. Piacentini Cattaneo, Milano, Springer, 2006

Crittografia: con elementi di teoria dei codici Wade Trappe, Lawrence C. Washington, Milano, Pearson Paravia Bruno Mondadori, 2009

RFC 1510, J.Khol, C. Neuman, Digital Equipment corporation, ISI, Settembre 1993

Sitografia

http://zeroshell.net/kerberos/
http://www.kerberos.org/
http://gost.isi.edu/publications/kerberos-neuman-tso.html
http://web.mit.edu/kerberos/
http://www.di.unisa.it/ ads/corso-security/www/CORSO-9900/unix/cap7.htm
http://www.itl.nist.gov/fipspubs/fip46-2.htm
http://www.di.unisa.it/ ads/corso-security/www/CORSO-0001/kerberos/cap2-5.html
http://www.di.unisa.it/professori/masucci/sicurezza0809/jdes/feistel.html
http://mccltd.net/blog/?p=1053
http://www.ietf.org/rfc/rfc1510.txt

Ringraziamenti

Desidero ringraziare il Professor Davide Aliffi per la disponibilità per l'autentico interesse dimostrati durante la preparazione della tesi.

Vorrei ringraziare la mia incasinata famiglia che, mi ha supportata e sopportata con costanza nell'avventura universitaria fino a questo momento.

Grazie infine agli amici, quelli veri, e a una persona che sto scoprendo giorno per giorno, per il loro ascolto, la loro comprensione, il loro esserci e per quel pizzico di leggerezza e profondità che sanno donarmi, rendendomi migliore, e spingendomi continuamente a cercare nuove domande.