

William Stallings

Sicurezza delle reti

Applicazioni e standard

Terza edizione



Copyright © 2007 Paravia Bruno Mondadori Editori
E-mail: hpeitalia@pearson.com
Web: <http://hpe.pearsoned.it>

Authorized translation from the English language edition, entitled NETWORK SECURITY ESSENTIALS: APPLICATIONS AND STANDARDS, 3rd Edition by STALLINGS, WILLIAM, published by Pearson Education, Inc, publishing as Prentice Hall, Copyright © 2007.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Italian language edition published by Paravia Bruno Mondadori Editori, Copyright © 2007.

Le informazioni contenute in questo libro sono state verificate e documentate con la massima cura possibile. Nessuna responsabilità derivante dal loro utilizzo potrà venire imputata agli Autori, a Paravia Bruno Mondadori Editori o a ogni persona e società coinvolta nella creazione, produzione e distribuzione di questo libro.

I diritti di riproduzione e di memorizzazione elettronica totale e parziale con qualsiasi mezzo, compresi i microfilm e le copie fotostatiche, sono riservati per tutti i paesi.

LA FOTOCOPIATURA DEI LIBRI È UN REATO L'editore potrà concedere a pagamento l'autorizzazione a riprodurre una porzione non superiore a un decimo del presente volume. Le richieste di riproduzione vanno inoltrate ad AIDRO (Associazione Italiana per i Diritti di Riproduzione delle Opere dell'Ingegno), Via delle Erbe, 2 – 20121 Milano – Tel. e Fax 02/80.95.06 – e-mail: segreteria@aidro.com.

Traduzione e curatela: Sara Grilli
Copy-editing: Bruno Lanata
Impaginazione: Elisabetta Bozzi
Stampa: Tip.Le.Co. – San Bonico (PC)

Tutti i marchi citati nel testo sono di proprietà dei loro detentori.

978-88-7192-345-1

Printed in Italy

3^a edizione: aprile 2007

*Ad Antigone
mai monotona
né noiosa
sempre assennata*

S O M M A R I O

Prefazione	xv
Obiettivi	xv
A chi è rivolto questo libro	xv
Piano dell'opera	xvi
Materiale didattico di supporto	xvi
Servizi Internet per docenti e studenti	xvii
Progetti per l'insegnamento della sicurezza di rete	xvii
Novità nella terza edizione	xviii
Ringraziamenti	xviii

Capitolo 1 Introduzione

1.1 Le tendenze della sicurezza	3
1.2 Architettura di sicurezza OSI	6
1.3 Attacchi alla sicurezza	7
Attacchi passivi	7
Attacchi attivi	7
1.4 Servizi di sicurezza	11
Autenticazione	11
Controllo dell'accesso	11
Riservatezza dei dati	13
Integrità dei dati	13
Non-ripudio	13
Disponibilità del servizio	13
1.5 Meccanismi di sicurezza	15
1.6 Un modello per la sicurezza di rete	15
1.7 Standard Internet e Internet Society	19
Le organizzazioni di Internet e la pubblicazione degli RFC	19
Processo di standardizzazione	20
Categorie degli standard Internet	22
Altri tipi di RFC	22
1.8 Contenuti del libro	22
1.9 Letture consigliate	23
1.10 Internet e risorse Web	23
Siti web	23
Altri siti web	24
Newsgroup USENET	25
1.11 Domande di revisione ed esercizi	25
Domande di revisione	25
Esercizi	25

PARTE 1 CRITTOGRAFIA**Capitolo 2 Cifratura simmetrica e riservatezza dei messaggi**

Capitolo 2 Cifratura simmetrica e riservatezza dei messaggi	29	
2.1 Elementi di cripotografia simmetrica	29	
Crittografia	30	
Criptanalisi	31	
Struttura del cifrario di Feistel	33	
2.2 Algoritmi di cripotografia simmetrica a blocchi	35	
Data Encryption Standard	35	
Descrizione dell'algoritmo	36	
Robustezza di DES	36	
Triplo DES	38	
Advanced Encryption Standard	39	
2.3 Cifrari a flusso e RC4	43	
Struttura di un cifrario a flusso	43	
L'algoritmo RC4	45	
2.4 Modalità operative dei cifrari a blocchi	47	
Modalità cipher block chaining	49	
Modalità cipher feedback	50	
2.5 Posizionamento dei dispositivi di cripotografia	52	
2.6 Distribuzione delle chiavi	54	
2.7 Letture consigliate e siti web	57	
Siti web consigliati	57	
2.8 Domande di revisione ed esercizi	57	
Domande di revisione	57	
Esercizi	57	
Capitolo 3 Crittografia a chiave pubblica e autenticazione dei messaggi	61	
3.1 Approcci all'autenticazione dei messaggi	61	
Autenticazione basata su cripotografia convenzionale	61	
Autenticazione dei messaggi senza cripotografia del messaggio	62	
3.2 Funzioni hash sicure e HMAC	66	
Requisiti di una funzione hash sicura	66	
Funzioni hash semplici	67	
La funzione hash sicura SHA-1	69	
Altre funzioni hash sicure	72	
HMAC	73	
3.3 Principi di crittografia a chiave pubblica	76	
Struttura della cripotografia a chiave pubblica	76	
Applicazioni dei sistemi di crittografia a chiave pubblica	79	
Requisiti della crittografia a chiave pubblica	80	
3.4 Algoritmi di crittografia a chiave pubblica	80	
Algoritmo a chiave pubblica RSA	80	
Algoritmo Diffie-Hellman per lo scambio di chiavi	83	
Altri algoritmi di crittografia a chiave pubblica	87	
3.5 Firme digitali	88	
3.6 Gestione delle chiavi	89	
Certificati a chiave pubblica	89	
Distribuzione a chiave pubblica di chiavi segrete	89	
3.7 Letture consigliate e siti web	91	
Siti web consigliati	91	
3.8 Domande di revisione ed esercizi	91	
Domande di revisione	91	
Esercizi	92	

PARTE 2 APPLICAZIONI DI SICUREZZA DI RETE

Capitolo 4 Applicazioni di autenticazione	99	
4.1 Kerberos	99	
Motivazioni	100	
Versione 4 di Kerberos	101	
Versione 5 di Kerberos	112	
4.2 Servizio di autenticazione X.509	119	
Certificati	120	
Procedure di autenticazione	124	
Versione 3 di X.509	126	
4.3 Infrastruttura per la chiave pubblica	129	
Funzioni di gestione della PKIX	130	
4.4 Letture consigliate e siti web	131	
Siti web consigliati	131	
4.5 Domande di revisione ed esercizi	132	
Domande di revisione	132	
Esercizi	132	
Appendice 4A Tecniche di cripotografia di Kerberos	133	
Trasformazione da password a chiave	133	

Capitolo 5 Sicurezza della posta elettronica

Capitolo 5 Sicurezza della posta elettronica	137	
5.1 Pretty good privacy	137	
Notazione	138	
Descrizione operativa	139	
Chiavi crittografiche e keyring	144	
Gestione delle chiavi pubbliche	153	
5.2 S/MIME	159	
RFC 822	159	
MIME	160	
Funzionalità di S/MIME	165	
Messaggi S/MIME	169	
Elaborazione dei certificati in S/MIME	173	
Servizi di sicurezza potenziati	176	
Siti web consigliati	176	

5.3 Domande di revisione ed esercizi	177
Domande di revisione	177
Esercizi	177
Appendice 5A Compressione dei dati mediante ZIP	178
Algoritmo di compressione	179
Algoritmo di decompressione	180
Appendice 5B Conversione radix-64	180
Appendice 5C Generazione di numeri casuali in PGP	182
Numeri casuali veri e propri	182
Numeri pseudocasuali	182
Capitolo 6 Sicurezza IP	185
6.1 Sicurezza IP: descrizione generale	185
Applicazioni di IPSec	185
Vantaggi di IPSec	186
Applicazioni di instradamento	188
6.2 Architettura di sicurezza IP	188
Documenti IPSec	188
Servizi IPSec	190
Associazioni di sicurezza	191
Modalità trasporto e tunnel	193
6.3 Intestazione di autenticazione	195
Servizio anti-replay	196
Valore per la verifica di integrità	197
Modalità trasporto e tunnel	198
6.4 Encapsulating security payload (ESP)	200
Formato di ESP	200
Algoritmi di cifratura e autenticazione	201
Completamento	202
Modalità trasporto e tunnel	202
6.5 Combinazione di più SA	206
Autenticazione e riservatezza	206
6.6 Gestione delle chiavi	209
ISAKMP	214
6.7 Letture consigliate e siti web	220
Siti web consigliati	220
6.8 Domande di revisione ed esercizi	220
Domande di revisione	220
Esercizi	221
Appendice 6A Internetworking e protocolli Internet	222
Ruolo di un protocollo Internet	222
IPv4	224
IPv6	226

Capitolo 7 Sicurezza web	231
7.1 Considerazioni sulla sicurezza web	231
Minacce alla sicurezza web	232
Approcci alla sicurezza del traffico web	232
7.2 Secure socket layer (SSL) e transport layer security (TLS)	234
Architettura SSL	234
Protocollo Record di SSL	236
Protocollo Change cipher spec	240
Protocollo Alert	240
Protocollo Handshake	241
Elaborazioni crittografiche	247
TLS	248
7.3 Secure electronic transaction (SET)	253
Descrizione generale di SET	254
Caratteristiche rilevanti di SET	255
Doppia firma	258
Elaborazione del pagamento	259
7.4 Letture consigliate e siti web	266
Siti web consigliati	266
7.5 Domande di revisione ed esercizi	266
Domande di revisione	266
Esercizi	266
Capitolo 8 Sicurezza della gestione di rete	269
8.1 Concetti base di SNMP	270
Architettura di gestione di rete	270
Architettura del protocollo di gestione di rete	271
Proxy	273
SNMPv2	275
8.2 Funzionalità di comunità di SNMPv1	278
Comunità e nomi di comunità	278
Servizio di autenticazione	279
Politica di accesso	279
Servizio proxy	280
8.3 SNMPv3	281
Architettura SNMP	282
Elaborazione di messaggi e modello di sicurezza utente	290
Controllo dell'accesso basato su viste	302
8.4 Letture consigliate e siti web	307
Siti web consigliati	307
8.5 Domande di revisione ed esercizi	307
Domande di revisione	307
Esercizi	307

PARTE 3 SICUREZZA DI SISTEMA**Capitolo 9 Intrusioni**

9.1	Intrusioni	313
	Tecniche di intrusione	313
9.2	Rilevazione delle intrusioni	315
	Record di audit	317
	Rilevazione statistica delle anomalie	319
	Rilevazione delle intrusioni basata su regole	321
	La Base-rate Fallacy	324
	Rilevazione distribuita delle intrusioni	326
	Honeypot	326
	Formato per lo scambio delle informazioni di rilevazione delle intrusioni	329
9.3	Gestione delle Password	330
	Protezione delle password	330
	Strategie di selezione delle password	335
9.4	Letture consigliate e siti web	340
	Siti web consigliati	340
9.5	Domande di revisione ed esercizi	341
	Domande di revisione	341
	Esercizi	341
Appendice 9A	La Base-Rate Fallacy	343
	Probabilità condizionale e indipendenza	343
	Teorema di Bayes	343
	Dimostrazione della Base-Rate Fallacy	344
		345

Capitolo 10 Software dolosi

10.1	Virus e minacce correlate	347
	Programmi dolosi	347
	Natura dei virus	347
	Tipologie di virus	350
	Virus delle macro	353
	Virus della posta elettronica	355
	Worm	355
	Stato della tecnologia degli worm	356
10.2	Rimedi contro i virus	358
	Approcci antivirus	359
	Tecniche antivirus di tipo avanzato	359
	Software a blocco del comportamento (behavior-blocking)	360
10.3	Attacchi di negazione del servizio distribuiti	363
	Descrizione degli attacchi DDoS	364
	Costruzione di una rete di attacco	364
	Rimedi per gli attacchi DDoS	368
10.4	Letture consigliate e siti web	369
	Siti web consigliati	369
10.5	Domande di revisione ed esercizi	370
	Domande di revisione	370
	Esercizi	370

311

Capitolo 11 Firewall

11.1	Principi di progettazione di firewall	371
	Caratteristiche dei firewall	371
	Tipologie di firewall	372
	Configurazione dei firewall	373
11.2	Sistemi trusted	382
	Controllo dell'accesso ai dati	384
	Concetto di sistema trusted	386
	Difesa dai cavalli di Troia	388
11.3	Criteri comuni per la valutazione della sicurezza	390
	Requisiti	390
	Profili e obiettivi	392
11.4	Letture consigliate e siti web	395
	Siti web consigliati	395
11.5	Domande di revisione ed esercizi	395
	Domande di revisione	395
	Esercizi	396

Appendice A Aspetti della teoria dei numeri

A.1	Numeri primi e relativamente primi	397
	Divisori	397
	Numeri primi	398
	Numeri relativamente primi	398
A.2	Aritmetica Modulare	399

Appendice B Progetti per l'insegnamento della sicurezza di rete

B.1	Progetti di ricerca	401
B.2	Progetti di programmazione	401
B.3	Esercizi di laboratorio	402
B.4	Compiti scritti	402
B.5	Compiti di lettura/relazione	402

Glossario**Bibliografia****Indice analitico**

PREFAZIONE

In quest'epoca di connettività elettronica globale, di virus e hacker, di spionaggio e frodi elettroniche, la sicurezza di rete ha assunto importanza crescente. Due tendenze hanno contribuito a rendere di interesse vitale gli argomenti di questo libro. Prima di tutto, la crescita esplosiva dei sistemi di elaborazione e delle loro interconnessioni attraverso le reti di comunicazione ha aumentato il livello di dipendenza delle organizzazioni e dei singoli utenti dalle informazioni memorizzate e scambiate su questi sistemi. Ciò, a sua volta, ha portato a un'accresciuta consapevolezza della necessità di proteggere dati e risorse da rilascio non autorizzato, di garantire autenticità di dati e messaggi e di proteggere i sistemi da attacchi che sfruttano le reti di comunicazione. Inoltre, lo sviluppo di crittografia e sicurezza di rete ha portato alla creazione di applicazioni utili e facilmente disponibili per l'implementazione della sicurezza.

Obiettivi

Scopo di questo libro è fornire un'utile rassegna delle applicazioni e degli standard relativi alla sicurezza di rete. L'enfasi è posta sulle applicazioni più diffusamente utilizzate in Internet, sulle reti aziendali e sugli standard di ampia diffusione, con particolare riferimento agli standard Internet.

A chi è rivolto questo libro

Il libro è rivolto a un pubblico sia accademico che professionale. Come libro di testo, è destinato a corsi di laurea in informatica, ingegneria informatica e ingegneria elettronica, per i corsi di sicurezza di rete.

Il libro può essere utilizzato anche come testo base di riferimento o di auto-apprendimento per professionisti del settore e manager.

Piano dell'opera

I contenuti sono suddivisi in tre parti.

- **Prima parte – Crittografia.** Breve rassegna degli algoritmi e dei protocolli di crittografia alla base delle applicazioni di sicurezza di rete, fra cui cifratura, funzioni hash, firme digitali e scambio di chiavi.
- **Seconda parte – Applicazioni di sicurezza di rete.** Tratta importanti strumenti e applicazioni sulla sicurezza di rete, fra cui Kerberos, certificati X.509v3, PGP, S/MIME, IP Security, SSL/TLS, SET e SNMPv3.
- **Terza parte – Sicurezza di sistema.** Dedicata a problematiche di sicurezza a livello di sistema, tra cui le minacce e le contromisure per intrusi e virus e l'impiego di firewall e sistemi trusted.

Ciascun capitolo è corredata da domande di revisione ed esercizi, letture consigliate e link a siti web su cui approfondire i vari argomenti. All'inizio di ciascuna parte, inoltre, troverete un dettagliato riassunto dei contenuti dei capitoli che la compongono.

Materiale didattico di supporto

I docenti che adottano il libro potranno accedere a supplementi didattici (in lingua inglese).

- **Solutions manual:** le soluzioni alle domande di revisione e agli esercizi posti al termine di ciascun capitolo.
- **Slide in PowerPoint®:** set completo delle slide suddivise per capitoli, utile per le lezioni in aula.
- **File PDF:** riproduzione di tutte le figure e tabelle presenti nel libro.
- **Projects manual:** progetti didattici di supporto ai docenti per l'insegnamento della sicurezza di rete.

Per accedere a questi supplementi, i docenti possono collegarsi al sito dedicato al libro, o contattare i rappresentanti Pearson Education.

Sul sito web dedicato al libro, i docenti potranno trovare anche:

- link a siti web di altri corsi sulla sicurezza in cui viene adottato questo testo;
- informazioni per iscriversi a una mailing list dedicata ai docenti della materia.

Servizi Internet per docenti e studenti

Docenti e studenti possono fare riferimento alla pagina web di supporto per questo libro. La pagina comprende riferimenti a siti web di particolare interesse, lucidi in formato PDF (Adobe Acrobat) e slide in PowerPoint®. La pagina web è disponibile all'indirizzo:

WilliamStallings.com/NetSec/NetSec3e.html

È stata approntata una lista di posta elettronica in modo che i docenti che utilizzano questo testo possano scambiare informazioni fra loro e con l'autore. Potranno anche segnalare eventuali errori o visionare la pagina degli *errata corrigere* sul sito WilliamStallings.com. Studenti di informatica e professionisti potranno inoltre trovare documenti, informazioni e link utili anche sul sito

WilliamStallings.com/StudentSupport.html

Progetti per l'insegnamento della sicurezza di rete

Per molti docenti, una componente importante di un corso sulla sicurezza e sulla crittografia è rappresentata dai progetti attraverso i quali lo studente può maturare esperienze sul campo e rafforzare così i concetti appresi. Per questo oltre al manuale delle soluzioni, è stato realizzato come ausilio didattico al libro un *Projects manual* che comprende una serie di suggerimenti inerenti i progetti da assegnare agli studenti, relativi a un ampio spettro di argomenti trattati nel testo.

- **Progetti di ricerca.** Esercitazioni che abituano lo studente a effettuare ricerche web su specifici argomenti e alla stesura del relativo report.
- **Progetti applicativi.** Esercitazioni di programmazione che coprono un ampio spettro di argomenti e che possono essere realizzate in un qualunque linguaggio adatto allo scopo su qualunque piattaforma.
- **Esercizi di laboratorio.** Una serie di progetti di programmazione e di sperimentazione su alcuni concetti presentati nel testo.
- **Compiti scritti.** Un insieme di suggerimenti, organizzati per capitolo.
- **Progetti compilativi.** Elenco di lavori di letteratura, uno per ciascun capitolo, che possono essere assegnati per la lettura e la redazione di un breve report.

Per informazioni più dettagliate si veda l'Appendice B.

Novità nella terza edizione

Negli ultimi tre anni, da quando è stata pubblicata la seconda edizione di questo libro, la sicurezza di rete ha subito continue innovazioni e miglioramenti. In questa nuova edizione si è cercato di raccogliere questi cambiamenti, pur continuando a coprire, nel modo più ampio possibile, tutti gli argomenti relativi a questo campo. All'inizio di questo processo di revisione, alcuni docenti di questa materia hanno esaminato dettagliatamente la seconda edizione di questo testo, inoltre i singoli capitoli sono stati controllati da vari professionisti del campo. Il risultato è stato un aumento della chiarezza e un ridimensionamento in molti punti del testo, oltre al miglioramento delle figure. Sono stati aggiunti anche un gran numero di nuovi esercizi derivati da esperienze reali, testati sul campo.

Oltre a questi raffinamenti per migliorare l'approccio didattico e la semplicità di utilizzo, sono stati apportati alcuni cambiamenti sostanziali. I più importanti comprendono:

- **Cifrari a flusso:** usati in numerosi protocolli e applicazioni di sicurezza di rete, sono trattati nella terza edizione, in particolare viene descritto RC4, che è l'algoritmo più diffuso.
- **Infrastruttura per la chiave pubblica (PKI, public key infrastructure):** importante argomento introdotto nella terza edizione.
- **Criteri comuni per la valutazione della sicurezza:** i criteri comuni sono diventati una struttura internazionalmente impiegata per esprimere i requisiti di sicurezza e valutare i prodotti e le implementazioni.

Inoltre, molti argomenti del libro sono stati aggiornati e revisionati.

Ringraziamenti

Questa nuova edizione ha beneficiato della revisione di molte persone, che hanno generosamente dedicato a questo libro il loro tempo e la loro esperienza, rileggendo, *in toto* o in parte, il manoscritto.

Le seguenti persone hanno contribuito all'aggiornamento e ai nuovi esercizi di questa edizione: Joshua Brandon Holden (Rose-Hulman Institute of Technology), Kris Gaj (George Mason University) e James Muir (University of Waterloo).

Sanjay Rao e Ruben Torres della Purdue University hanno sviluppato gli esercizi di laboratorio presenti nel supplemento web. Le seguenti persone hanno fornito materiale per i progetti di laboratorio (anch'essi presenti sul Web): Henning Schulzrinne (Columbia University), Cetin Kaya Koç (Oregon State University) e David Balenson (Trusted Information Systems and George Washington University).

Desidero quindi ringraziare tutte le persone responsabili della pubblicazione di questo testo, che hanno fatto un eccellente lavoro. Mi riferisco a tutto lo staff di Prentice Hall, in particolare al Production Manager Rose Kernan, al mio editor Tracy Dunkelberger e alle sue assistenti Christianna Lee e Carole Snyder. Ringrazio anche Patricia M. Daly che si è occupata del copy-editing.

Capitolo 1

Introduzione

Negli ultimi decenni, i requisiti di **sicurezza delle informazioni** all'interno di un'organizzazione hanno subito due principali cambiamenti. Prima della diffusione degli strumenti di elaborazione dati, la sicurezza delle informazioni considerate di particolare valore per un'organizzazione (*dati sensibili*) era fornita principalmente da strumenti di tipo fisico e amministrativo quali armadietti di archiviazione con chiusura a combinazione per la conservazione di documenti riservati o colloqui personali riguardanti il processo di assunzione.

Con l'introduzione degli elaboratori, è diventata evidente la necessità di strumenti automatici per la protezione dei file e delle altre informazioni ivi memorizzate. Questo è vero in particolare per i sistemi condivisi, come un sistema time sharing (a condivisione di tempo). Necessità che si rivela particolarmente cruciale per quei sistemi cui si può accedere tramite rete telefonica pubblica, rete dati o da Internet. Si indica con il termine generico di **sicurezza dei sistemi di elaborazione** (*computer security*) l'insieme degli strumenti progettati al fine di proteggere i dati e contrastare i pirati informatici (*hacker*).

Il secondo principale cambiamento riguarda l'introduzione dei sistemi distribuiti e l'uso delle tecnologie di rete e di comunicazione per il trasporto dati tra terminali utente ed elaboratori e tra elaboratori.

Per proteggere i dati durante la loro trasmissione, sono necessarie misure di sicurezza a livello di rete. Di fatto, il termine **sicurezza di rete** (*network security*) è talvolta ingannevole, in quanto praticamente tutte le organizzazioni aziendali, governative e accademiche connettono i loro sistemi di elaborazione dati mediante un insieme di reti tra loro interconnesse. A tale insieme si fa spesso riferimento come a una *internet*¹ da cui la locuzione **sicurezza-internet** (*internet security*).

Non esistono confini ben definiti fra queste due forme di sicurezza. Ad esempio, uno fra i più noti tipi di attacco nei sistemi informativi è il virus del calcolatore. Un virus può essere fisicamente introdotto all'interno di un sistema attraverso supporti esterni. I virus possono giungere anche attraverso la rete. In entrambi i casi, una volta che il virus è residente nel sistema di elaborazione, sono necessari strumenti interni di computer security per poterne effettuare la rilevazione e procedere al ripristino del sistema.

¹ Si usa il termine *internet*, con la "i" minuscola, per riferirsi a qualsiasi insieme di reti interconnesse. Un'intranet aziendale ne è un esempio. Internet – con la "I" maiuscola – può essere una delle funzionalità utilizzate da un'organizzazione per costruire la propria *internet*.

Questo libro si focalizza sulla sicurezza delle reti di comunicazione, che comprende l'insieme di misure volte a impedire, prevenire, rilevare e correggere violazioni alla sicurezza, che coinvolgono la trasmissione di informazioni. Questa definizione è ad ampio spettro e copre una serie di possibilità. Per dare l'idea dei temi trattati in questo libro, si considerino i seguenti esempi di violazioni alla sicurezza.

1. L'utente A trasmette un file all'utente B. Il file contiene informazioni sensibili (ad esempio registrazioni di pagamenti) che devono essere protette da divulgazione non autorizzata. L'utente C, che non è autorizzato a leggere il file, è in grado di monitorare la trasmissione e di effettuare una copia del file in fase di trasmissione.
2. L'amministratore di rete, D, trasmette un messaggio al computer, E, di cui è gestore. Questo messaggio dà istruzione ad E affinché aggiorni un file delle autorizzazioni aggiungendo le identità di un certo numero di nuovi utenti autorizzati ad accedere ad E. L'utente F intercetta il messaggio, ne modifica il contenuto, aggiungendo o cancellando voci, e poi inoltra il messaggio ad E, il quale lo accetta come se provenisse dal gestore D, e provvede a modificare di conseguenza il proprio file delle autorizzazioni.
3. Invece di intercettare il messaggio, l'utente F crea un proprio messaggio contenente le voci (di modifica) desiderate e lo trasmette ad E, come se fosse stato inoltrato dal gestore D. L'elaboratore E accetta il messaggio come se provenisse dal gestore D e provvede a modificare di conseguenza il proprio file delle autorizzazioni.
4. Un impiegato viene licenziato senza preavviso. Il direttore del personale invia un messaggio al server per invalidare l'account dell'impiegato. Terminata l'operazione di invalidazione, il server deve inviare una notifica all'impiegato a conferma dell'avvenuta operazione. L'impiegato è in grado di intercettare il messaggio inviato dal direttore e di posticiparne l'invio al server per il tempo sufficiente a consentirgli un ultimo accesso per recuperare informazioni sensibili. Il messaggio viene poi inoltrato, l'operazione eseguita e la notifica inviata. L'azione dell'impiegato potrebbe passare inosservata anche per molto tempo.
5. Un cliente invia a un agente di cambio un messaggio contenente istruzioni per l'esecuzione di un certo numero di transazioni. Successivamente, gli investimenti perdono di valore e il cliente nega di avere inviato tale messaggio.

Anche se questa lista non è certamente esaustiva di tutte le possibili tipologie di violazioni, contribuisce a illustrare l'ambito di interesse della sicurezza di rete, argomento a un tempo affascinante e complesso, per svariati motivi.

1. La sicurezza che coinvolge la comunicazione e le reti di calcolatori non è così semplice come potrebbe apparire a un principiante. Infatti, gran parte dei principali requisiti alla base dei servizi di sicurezza possono essere formulati mediante singoli termini auto-esplicativi: riservatezza, autenticazione, non-ripudio, integrità. Ma i meccanismi impiegati risultano piuttosto complessi, e la loro comprensione può richiedere ragionamenti piuttosto sottili.
2. Nello sviluppo di un particolare meccanismo o algoritmo di sicurezza, occorre sempre considerare i potenziali attacchi che potrebbe subire. In molti casi, gli attacchi che hanno successo sono progettati affrontando il problema da un altro punto di vista, così da sfruttare un punto debole, non previsto nel meccanismo di sicurezza.

3. Per le motivazioni espresse al punto 2, le procedure utilizzate per fornire specifici servizi di sicurezza risultano sovente tutt'altro che intuitive: molte volta la descrizione di uno specifico requisito non lascia intuire come in effetti siano necessarie misure estremamente elaborate. Le misure impiegate acquistano un senso solo quando si considerano le diverse strategie di attacco nel loro insieme.
4. Dopo aver progettato svariati meccanismi di sicurezza, è necessario decidere dove utilizzarli. Questo vale sia dal punto di vista della localizzazione fisica (per esempio, in quali punti di una rete sono necessari) sia dal punto di vista logico (per esempio, a quale livello o a quali livelli di un'architettura, come quella TCP/IP, devono essere collocati).
5. Generalmente, i meccanismi di sicurezza coinvolgono più di un singolo algoritmo o protocollo. Inoltre, richiedono che i partecipanti siano in possesso di informazioni segrete (per esempio, una chiave di cifratura), e ciò induce problematiche relative alla creazione, distribuzione e protezione di tali informazioni. Ancora, il compito di sviluppare i meccanismi di sicurezza potrebbe essere complicato dal comportamento dei protocolli di comunicazione su cui i meccanismi di sicurezza si appoggiano. Così, se il corretto funzionamento di un meccanismo di sicurezza richiede l'impostazione di limitazioni al tempo di transito di un messaggio da un mittente a un destinatario, un qualsiasi protocollo o una rete che introduca ritardi variabili e non prevedibili potrebbe vanificare tali limitazioni.

Occorre, pertanto, considerare molteplici aspetti. Questo capitolo fornisce una panoramica generale delle problematiche che saranno approfondite nel prosieguo del libro. Il capitolo inizia trattando le tipologie di attacchi che rendono necessari servizi e meccanismi di sicurezza di rete. Successivamente, viene descritto un modello generale in cui collocare i diversi servizi e meccanismi di sicurezza.

1.1 Le tendenze della sicurezza

Nel 1994, l'Internet Architecture Board (IAB) rilasciò la relazione "La sicurezza nell'architettura di Internet" (RFC 1636). Il rapporto esprimeva la concordanza di opinioni sul fatto che Internet necessitasse di una maggiore e migliore sicurezza, oltre che a identificare le aree chiave per i meccanismi di sicurezza. Tra queste figurava l'esigenza di rendere sicura l'infrastruttura di rete da monitoraggi e da controlli non autorizzati del traffico di rete, e il bisogno di rendere sicuro il traffico tra utenti finali, utilizzando meccanismi di autenticazione e crittografia.

Queste preoccupazioni sono pienamente giustificate. Come conferma, si considerino le tendenze riportate dal Centro di coordinamento (CC) del Computer Emergency Response Team (CERT). La Figura 1.1a mostra l'andamento delle vulnerabilità legate a Internet riportate dal CERT su un periodo di 10 anni. Queste comprendono sia i punti deboli di sicurezza nei sistemi operativi dei computer collegati (ad esempio, Windows o Linux), sia le vulnerabilità dei router di Internet e degli altri dispositivi di rete. La Figura 1.1b mostra il numero di incidenti legati alla sicurezza, riportati dal CERT. Questi includono gli attacchi di negazione del servizio (*denial of service*); gli IP spoofing, nei quali un intruso crea pac-

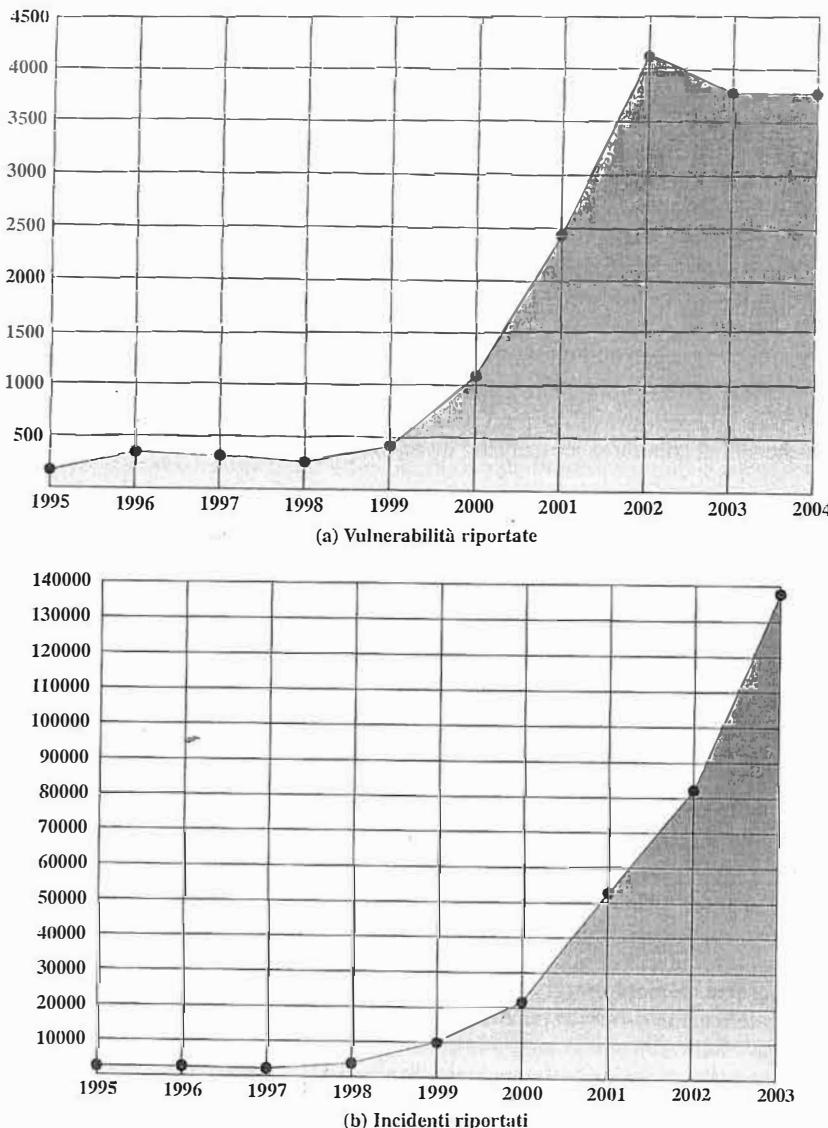


Figura 1.1 Statistiche del CERT.

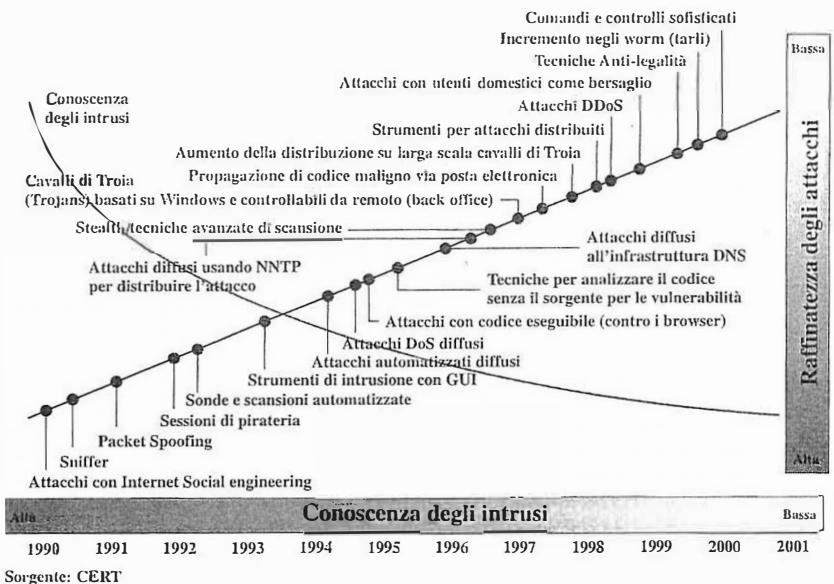


Figura 1.2 Tendenze nella raffinatezza degli attacchi e nella conoscenza degli intrusi.

chetti IP con un indirizzo IP falso e sfrutta le applicazioni che usano l'autenticazione basata su IP, e varie forme di intercettazione e cattura dei pacchetti, nei quali gli attaccanti leggono le informazioni trasmesse, comprese le informazioni di accesso e il contenuto dei database.

Col trascorrere del tempo, gli attacchi su Internet e ai sistemi collegati a Internet sono diventati sempre più sofisticati, mentre la quantità di capacità e conoscenze necessarie per lanciare un attacco sono diminuite (Figura 1.2). È aumentata l'automatizzazione degli attacchi e la possibilità di arrecare gravi danni.

L'incremento negli attacchi è coinciso con lo sviluppo dell'uso di Internet e con l'aumento complessità dei protocolli, delle applicazioni e di Internet stessa. Sempre più infrastrutture critiche dipendono da Internet per funzionare; gli utenti fanno sempre maggiore affidamento alla sicurezza di Internet, della posta elettronica, del Web e delle applicazioni basate sul Web. In questo modo, risultano necessari sia un'ampia gamma di tecnologie sia strumenti per combattere la crescente minaccia. A livello base, gli algoritmi crittografici, per la riservatezza e l'autenticazione, assumono una maggiore importanza. Allo stesso modo, i progettisti hanno bisogno di focalizzarsi su protocolli basati su Internet, e sulle vulnerabilità dei sistemi operativi e delle applicazioni collegate. Questo libro esamina tutte queste aree tecnologiche.

1.2 Architettura di sicurezza OSI

Per stimare efficacemente le necessità di sicurezza di un'organizzazione, valutare e scegliere fra le diverse politiche e i vari prodotti, il responsabile della sicurezza ha bisogno di un metodo sistematico per la definizione dei requisiti di sicurezza e per l'analisi degli approcci da adottare per soddisfare tali requisiti. Se questo risulta abbastanza difficile in un ambiente di elaborazione dei dati centralizzato, il problema si aggrava con l'utilizzo di una rete locale o geografica.

La raccomandazione ITU-T² X.800, "Architettura di sicurezza per OSI", definisce questo tipo di approccio sistematico. L'architettura di sicurezza OSI è utile agli amministratori come modo per organizzare le attività per fornire sicurezza. Inoltre, dato che questa architettura è stata sviluppata come standard internazionale, i fornitori di computer e di sistemi di comunicazione hanno incrementato le caratteristiche di sicurezza dei loro prodotti e servizi, in modo da far riferimento a questa definizione strutturata di servizi e meccanismi.

Per i nostri scopi, l'architettura di sicurezza OSI fornisce, se riassunta, un'utile visione di molti concetti affrontati in questo libro. L'architettura di sicurezza OSI si focalizza su attacchi, meccanismi e servizi di sicurezza. Questi possono essere sinteticamente definiti come:

- **attacchi alla sicurezza:** qualunque azione che compromette la sicurezza delle informazioni possedute da un'organizzazione;
- **meccanismi di sicurezza:** progettati per rilevare, prevenire o porre rimedio agli effetti di un attacco alla sicurezza;
- **servizi di sicurezza:** realizzano la sicurezza dei sistemi di elaborazione e di trasmissione dati di un'organizzazione. Tali servizi hanno lo scopo di contrastare gli attacchi alla sicurezza facendo uso di uno o più meccanismi.

Nella letteratura, i termini **minaccia** e **attacco** sono comunemente usati per indicare più o meno la stessa cosa. Nella Tabella 1.1 sono riportate le definizioni prese dalla RFC 2828, "Glossario della sicurezza di Internet".

Minaccia.

Potenziale violazione di sicurezza, che diventa realtà quando si presentano le circostanze, le capacità, le azioni o gli eventi che possono aprire una breccia nella sicurezza e infliggere danni. Vale a dire che una minaccia è un potenziale pericolo che potrebbe sfruttare una vulnerabilità.

Attacco.

Assalto al sistema di sicurezza che deriva da una minaccia intelligente; cioè un tentativo deliberato (specialmente nel senso di un metodo o di una tecnica) per eludere i servizi di sicurezza e violare le politiche di sicurezza di un sistema.

Tabella 1.1 Minacce e attacchi.

² L'International Telecommunication Union (ITU) nel settore di standardizzazione delle Telecomunicazioni (ITU-T) è una agenzia sponsorizzata dalle Nazioni Unite, che sviluppa standard, chiamati "raccomandazioni", che fanno riferimento alle telecomunicazioni e all'interconnessione di sistemi aperti (OSI).

1.3 Attacchi alla sicurezza

Un modo utile per classificare gli attacchi alla sicurezza, usato sia in X.800 sia nella RFC 2828, fa riferimento alle locuzioni **attacchi passivi** e **attacchi attivi**. I primi tentano di apprendere – o far uso di – informazioni dal sistema, ma non ne colpiscono le risorse. I secondi cercano di alterarne le risorse o pregiudicare il loro funzionamento.

Attacchi passivi

Gli attacchi passivi sono connessi al monitoraggio e all'intercettazione delle trasmissioni. Obiettivo dell'attaccante è carpire informazioni durante la trasmissione. Il rilascio del contenuto di un messaggio e l'analisi del traffico sono due tipologie di attacchi passivi.

Il **rilascio del contenuto di un messaggio** è di facile comprensione (Figura 1.3a). Una conversazione telefonica, un messaggio di posta elettronica, un file trasferito potrebbero contenere informazioni sensibili o riservate. Si vorrebbe impedire all'avversario di apprendere i contenuti di queste trasmissioni. Il secondo tipo di attacco passivo, **l'analisi del traffico**, è più sottile. Si supponga di disporre di un modo per mascherare il contenuto dei messaggi o altre informazioni di traffico così che gli avversari, anche se intercettassero il messaggio, non potrebbero estrarre l'informazione contenuta. La tecnica comunemente utilizzata per mascherare i contenuti è la cifratura. Pur disponendo della protezione assicurata dalla cifratura, un avversario potrebbe essere ancora in grado di osservare le tipologie di tali messaggi. L'avversario potrebbe determinare la posizione e l'identità degli host comunicanti ed osservare la frequenza e la lunghezza dei messaggi scambiati. Queste informazioni potrebbero essere utili per inferire la natura della comunicazione in corso.

Gli attacchi passivi sono molto difficili da individuare perché non implicano alcuna modifica dei dati. Tipicamente, il traffico dei messaggi è inviato e ricevuto in modo apparentemente normale e né il mittente né il destinatario si rendono conto che una terza parte ha letto il messaggio o osservato lo schema del traffico. Tuttavia, è possibile prevenire il successo di questi attacchi, di solito attraverso la crittografia. L'enfasi nell'affrontare questi attacchi, quindi, è sulla prevenzione più che sull'individuazione.

Attacchi attivi

Gli attacchi attivi implicano una qualche modifica al flusso dei dati o la creazione di un falso flusso e possono essere suddivisi in quattro categorie: masquerade, replay, modifica dei messaggi e negazione del servizio.

Un attacco **masquerade** si verifica quando un'entità finge di essere un'entità diversa (Figura 1.4a). Generalmente, comprende una delle altre tipologie di attacchi attivi. Ad esempio, è possibile rilevare e replicare sequenze di autenticazione dopo che ha avuto luogo una sequenza di autenticazione valida, consentendo così a un'entità autorizzata, ma che dispone di pochi privilegi, di ottenerne di aggiuntivi impersonando l'entità che li detiene.

Un attacco di **replay** coinvolge l'intercettazione passiva di dati e la successiva ritrasmissione degli stessi al fine di generare un effetto non autorizzato (Figura 1.4b).

La **modifica di messaggi** significa semplicemente che alcune porzioni di un messaggio legittimo vengono modificate, oppure che i messaggi sono ritardati o riordinati con lo scopo di produrre un effetto non autorizzato (Figura 1.4c). Ad esempio, un messaggio come

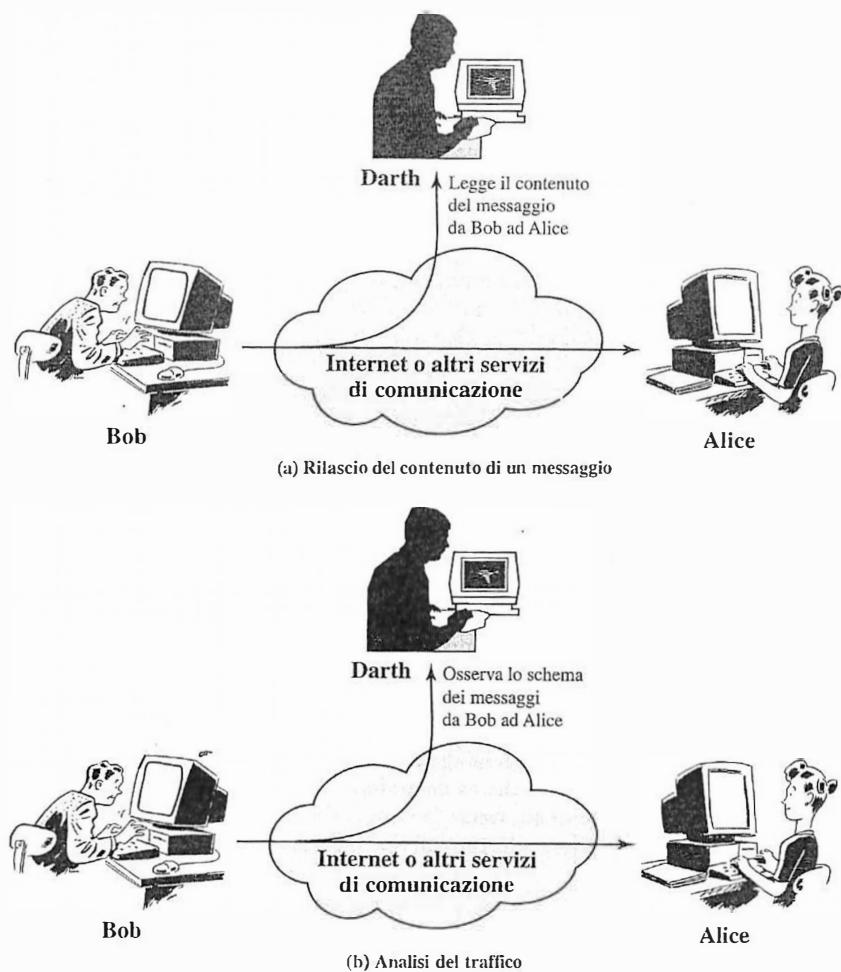


Figura 1.3 Attacchi passivi.

"Permetti a Paolo Rossi la lettura del file riservato *stipendi*" può essere modificato in "Permetti a Giovanni Bianchi la lettura del file riservato *stipendi*".

La **negazione del servizio** impedisce o inibisce il normale utilizzo (autorizzato) o la gestione delle funzionalità di comunicazione (Figura 1.4d). Questo attacco può avere un obiettivo specifico; ad esempio, un'entità può eliminare tutti i messaggi diretti a una particolare destinazione (come il servizio di verifica della sicurezza). Un'altra forma di negazione del servizio è l'interruzione di un'intera rete; ciò avviene mettendo fuori uso la rete o sovraccaricandola con messaggi per degradarne le prestazioni.

Gli attacchi attivi presentano caratteristiche opposte a quelli passivi. Questi ultimi sono difficili da rilevare ma sono disponibili misure per impedire che abbiano successo. Al contrario, è piuttosto difficile prevenire gli attacchi attivi, in quanto ciò richiederebbe una protezione completa di tutte le funzionalità di comunicazione e delle vie di accesso in ogni momento. Invece, l'obiettivo è quello di rilevare questi attacchi e ripristinare il sistema da qualunque interruzione o ritardo da essi causato. Poiché la rilevazione ha un effetto deterrente, può contribuire anche alla prevenzione.

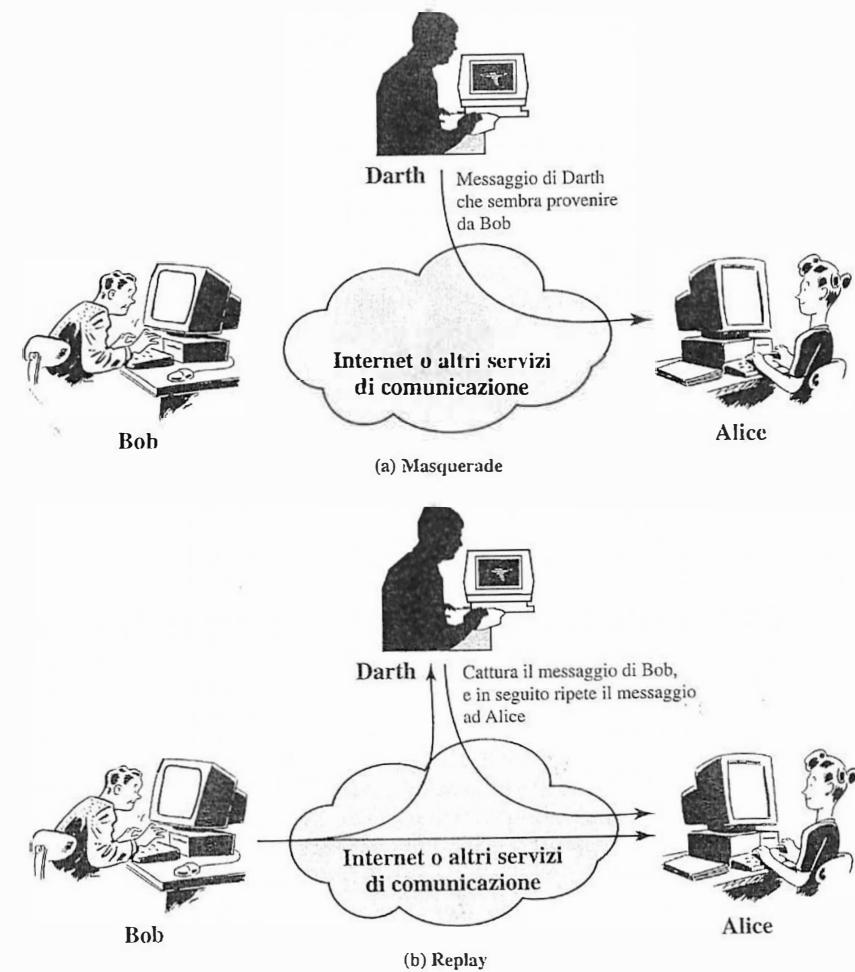


Figura 1.4 Attacchi attivi (continua).

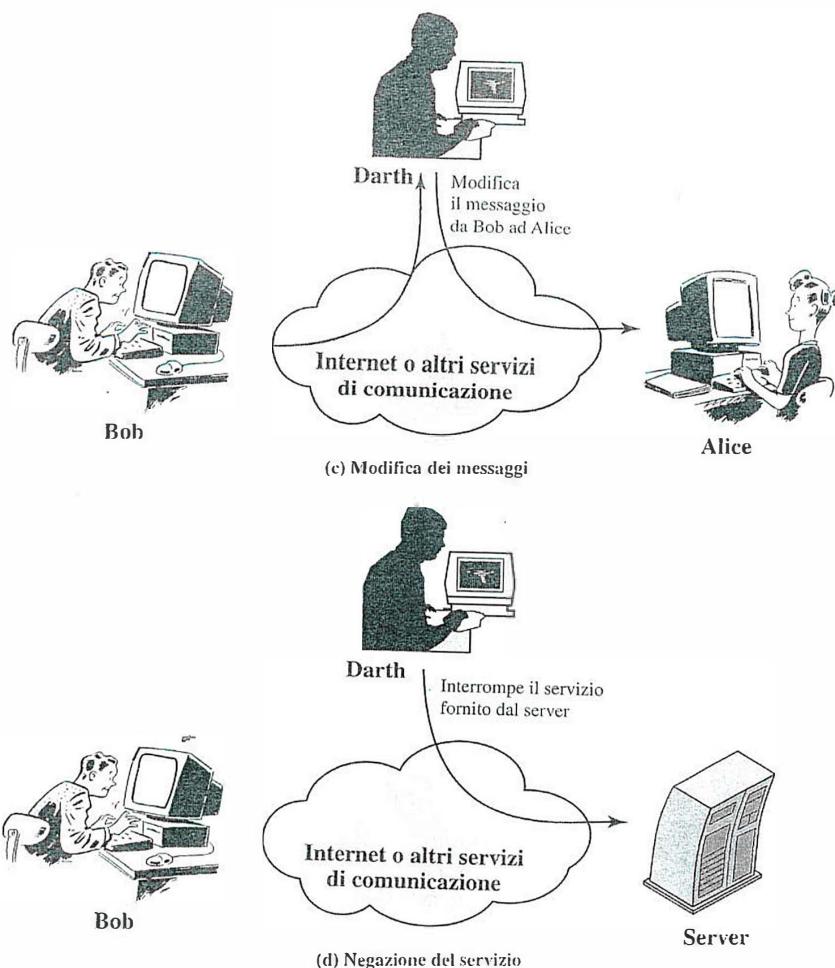


Figura 1.4 Attacchi attivi (segue).

1.4 Servizi di sicurezza

In X.800 un servizio di sicurezza viene definito come un servizio fornito da un livello protocollo di sistemi aperti, comunicanti tra loro, e che assicura un'adeguata sicurezza dei sistemi o dei trasferimenti di dati. Forse una definizione più chiara si può trovare nell'RFC 2828, che riporta la seguente definizione: un servizio di elaborazione o di comunicazione che è fornito da un sistema per dare uno specifico tipo di protezione alle risorse del sistema; i servizi di sicurezza implementano politiche di sicurezza e sono implementati da meccanismi di sicurezza.

In X.800 questi servizi sono divisi in cinque categorie e quattordici servizi specifici (Tabella 1.2). Verrà presa in considerazione nel seguito ciascuna categoria a turno³.

Autenticazione

Il servizio di autenticazione fornisce garanzia dell'autenticità di una comunicazione. In caso di un singolo messaggio, come ad esempio segnali di avviso o di allarme, la funzione del servizio di autenticazione è quella di assicurare il ricevente che il messaggio è stato effettivamente inviato dalla sorgente dichiarata. Nel caso di un'interazione in corso, come ad esempio una connessione tra un terminale e un host, vanno considerati due aspetti. Prima di tutto, al momento dell'inizio della connessione, il servizio assicura che le due entità siano autentiche (cioè che ciascuna sia effettivamente quello che sostiene di essere). Successivamente, il servizio deve assicurare che la connessione sia priva di interferenze che rendano possibile a una terza parte fingersi una delle due parti legittime (attacco masquerade) per scopi di trasmissione o ricezione non autorizzate.

Nel standard sono definiti due servizi specifici di autenticazione.

- **Autenticazione dell'entità paritaria.** Si preoccupa di avvalorare l'identità di un'entità paritaria in un'associazione. Due entità sono considerate paritarie se implementano lo stesso protocollo in sistemi diversi: ad esempio, due moduli TCP in due sistemi comunicanti. È fornita per essere usata durante la creazione o talvolta durante la fase di trasferimento dati di una connessione.
- **Autenticazione dell'origine dei dati.** Si preoccupa di confermare la sorgente di un'unità di dati. Non fornisce protezione contro la duplicazione o la modifica delle unità dati. Questo tipo di servizio supporta le applicazioni come la posta elettronica, dove non ci sono interazioni antecedenti tra le entità comunicanti.

Controllo dell'accesso

Nell'ambito della sicurezza di rete, il controllo dell'accesso è la capacità di limitare e controllare l'accesso a sistemi host e applicazioni attraverso canali di comunicazione. Per ottenere questo controllo, ogni entità che tenta di accedere deve prima di tutto essere identificata e autenticata, in modo che si possano applicare i diritti di accesso specificati per l'entità in questione.

³ Non esiste univocità di significato per molti termini utilizzati nella letteratura di sicurezza. Ad esempio, il termine integrità è talvolta utilizzato per fare riferimento a tutti gli aspetti di sicurezza delle informazioni. Il termine autenticazione viene qualche volta utilizzato per far riferimento sia alla verifica dell'identità sia alle varie funzioni elencate sotto l'integrità in questo capitolo. L'utilizzo qui è stato fatto in accordo sia con l'X.800 sia con la RFC 2828.

AUTENTICAZIONE	INTEGRITÀ DEI DATI
L'assicurazione che l'entità comunicante sia quella che dice di essere.	L'assicurazione che i dati ricevuti siano esattamente come quelli inviati dall'entità autorizzata (cioè non contengano modifiche, inserimenti, cancellazioni o ripetizioni).
CONTROLLO DEGLI ACCESSI	RISERVATEZZA DEI DATI
La prevenzione di un uso non autorizzato di una risorsa (cioè questo servizio controlla chi può avere accesso a una risorsa, in quali condizioni l'accesso può avvenire e cosa hanno il permesso di fare quelli che accedono alla risorsa).	La protezione dei dati da una divulgazione non autorizzata.
Riservatezza della connessione La protezione di tutti i dati utente sulla connessione.	Riservatezza in assenza di connessione La protezione di tutti i dati utente in un singolo blocco di dati.
Riservatezza selettiva per campi La confidenzialità dei campi selezionati all'interno dei dati utente su una connessione o in un singolo blocco di dati.	Riservatezza del flusso di traffico La protezione delle informazioni che potrebbero essere ricavate dell'osservazione dei flussi di traffico.
NON RIPUDIO	
Fornisce protezione contro la negazione, da parte di una delle entità coinvolte in una comunicazione, di aver partecipato a tutta o a parte della comunicazione.	
Non ripudio, origine La prova che il messaggio è stato mandato dalla parte specificata.	
Non ripudio, destinazione La prova che il messaggio è stato ricevuto dalla parte specificata.	

Tabella 1.2 Servizi di sicurezza (X.800).

Riservatezza dei dati

La riservatezza è la protezione dei dati trasmessi dagli attacchi passivi. Rispettando il contenuto della trasmissione dati, possono essere identificati vari livelli di protezione. Il servizio più generale protegge tutti i dati trasmessi tra due utenti in un certo periodo di tempo. Ad esempio, quando viene inizializzata una connessione TCP tra due sistemi, questa ampia protezione evita che qualsiasi dato utente trasmesso sulla connessione TCP venga reso pubblico. Possono essere definite forme più precise di questo servizio, che comprendono la protezione di un singolo messaggio o anche di campi specifici all'interno di un messaggio. Queste raffinatezze sono meno utili dell'approccio generale e potrebbero essere anche più complesse e costose da implementare.

L'altro aspetto della riservatezza è la protezione del flusso di traffico dall'analisi. Questo richiede che l'attaccante non possa osservare la sorgente e la destinazione, la frequenza, la lunghezza o altre caratteristiche del traffico su un servizio di comunicazione.

Integrità dei dati

Come per la riservatezza, l'integrità si può applicare a un flusso di messaggi, a un singolo messaggio, o a campi selezionati all'interno di un messaggio. Di nuovo, l'approccio più utile e semplice consiste nella protezione totale dei flussi.

Un servizio di integrità orientato alla connessione, che agisce a livello di flusso di messaggi, assicura che questi siano ricevuti come sono stati spediti, senza duplicazione, inserimento, modifica, riordinamento, repliche o distruzione dei dati. Quindi, un servizio di integrità orientato alla connessione considera sia modifiche del flusso di messaggi sia la negazione del servizio. D'altro canto, un servizio di integrità non orientato alla connessione, ovvero un servizio che tratta solo singoli messaggi senza tenere in considerazione alcun contesto più ampio, fornisce generalmente protezione solo rispetto a modifiche del messaggio.

È possibile distinguere servizi con e senza ripristino. Dato che il servizio di integrità è relativo ad attacchi attivi, risulta di maggiore interesse la rilevazione rispetto alla prevenzione. Nel caso in cui venga rilevata una violazione alla sicurezza, il servizio potrebbe limitarsi semplicemente a segnalarla, rendendo necessario l'utilizzo di uno specifico software o l'intervento di un operatore per effettuare il ripristino. Come vedremo successivamente, in alternativa sono disponibili meccanismi per effettuare il ripristino a seguito della perdita di integrità dei dati. La possibilità di incorporare meccanismi automatici di ripristino è, in generale, l'alternativa più interessante.

Non-ripudio

Obiettivo del non-ripudio è impedire che il mittente o il destinatario neghino che sia stato trasmesso un messaggio. Quindi, quando un messaggio viene inviato, il destinatario può provare che il messaggio è stato in effetti inviato dal presunto mittente. In modo analogo, al ricevimento di un messaggio, il mittente può provare che il messaggio è stato effettivamente ricevuto dal presunto destinatario.

Disponibilità del servizio

Sia in X.800 che in RFC 2828 si definisce la disponibilità come una proprietà di un sistema o di una risorsa del sistema, cui un'entità di un sistema autorizzato può accedere e può usare su richiesta, secondo le specifiche di funzionamento del sistema (cioè, un sistema è disponibile).

MECCANISMI DI SICUREZZA SPECIFICI	MECCANISMI DI SICUREZZA PERVERSIVI
Cifratura Possono essere incorporati in un livello protocollore appropriato per fornire alcuni dei servizi di sicurezza OSI.	Meccanismi che non sono specifici di un particolare servizio di sicurezza OSI o livello protocollore.
Firma digitale Un'aggiunta di dati in una trasformazione crittografica di un'unità di dati, che consente al ricevente dell'unità di dati di dimostrare l'autenticità della sorgente e l'integrità dell'unità di dati, oltre che di proteggerla dalla contraffazione (ad esempio, da parte del ricevitore)	Funzionalità fidate Ciò che viene percepito essere corretto nel rispetto di alcuni criteri (ad esempio, come stabilito da una politica di sicurezza).
Controllo degli accessi Una varietà di meccanismi che rafforzano i diritti di accesso alle risorse.	Etichetta di sicurezza La marcatura legata a una risorsa (che potrebbe essere un'unità dati) che stabilisce o indica gli attributi di sicurezza di quella risorsa.
Integrità dei dati Una varietà di meccanismi che assicurano l'integrità di un'unità di dati o di un flusso di unità di dati.	Individuazione degli eventi Individuazione di eventi rilevanti per la sicurezza.
Scambio di autenticazioni Un meccanismo volto a assicurare l'identità di un'entità per mezzo di uno scambio di informazioni.	Prove della verifica di sicurezza Dati raccolti che possono essere potenzialmente usati per facilitare una verifica di sicurezza, che è una revisione e un esame indipendente della documentazione e delle attività di sicurezza.
Traffico di riempimento L'inserimento di bit nei vuoli di un flusso di dati per rendere vani i tentativi di analisi del traffico.	Recupero di sicurezza Si occupa delle richieste provenienti dai meccanismi di sicurezza, come il modo di affrontare un evento o le funzioni di gestione, e si adopera per il recupero della sicurezza.
Controllo dell'instradamento Abilità la selezione di un particolare instradamento fisico sicuro per certi dati e consente il cambiamento degli instradamenti, specialmente quando si sospetta una violazione della sicurezza.	
Certificazione L'utilizzo di una terza parte fidata per assicurare certe proprietà di uno scambio di dati.	

Tabella 1.3 Meccanismi di sicurezza (X.800).

nibile se fornisce servizi, in conformità al progetto del sistema, ognualvolta gli utenti li richiedano). Vari tipi di attacchi possono causare la perdita e la riduzione della disponibilità. Alcuni di questi attacchi si prestano a contromisure automatizzate come, ad esempio, autenticazione e cifratura, mentre altri necessitano di un'azione fisica per impedire o recuperare la perdita di disponibilità di elementi di un sistema distribuito.

X.800 tratta la disponibilità come una proprietà che va associata a diversi servizi di sicurezza. Tuttavia, ha senso indicare specificatamente un servizio di disponibilità. Un servizio di disponibilità protegge un sistema assicurandone la sua disponibilità. Questo servizio mira a risolvere i problemi di sicurezza derivanti dagli attacchi di negazione del servizio. Dipende da una gestione e un controllo appropriati delle risorse del sistema e quindi fa affidamento sul servizio di controllo degli accessi e su altri servizi di sicurezza.

1.5 Meccanismi di sicurezza

Nella Tabella 1.3 sono elencati i meccanismi di sicurezza definiti in X.800. Come si può vedere, sono suddivisi in meccanismi implementati in un specifico livello protocollore e in quelli non specifici a un particolare livello protocollore o servizio di sicurezza.

Questi meccanismi saranno trattati nei successivi capitoli e non verranno analizzati ora, eccetto che commentare la definizione di cifratura.

In X.800 si distingue tra meccanismi di cifratura reversibili e quelli irreversibili. Un meccanismo di cifratura reversibile è semplicemente un algoritmo di crittografia che consente di cifrare i dati e successivamente decifrarli. I meccanismi di cifratura irreversibile includono gli algoritmi hash e i codici di autenticazione del messaggio, usati nella firma digitale e nelle applicazioni di autenticazione del messaggio.

Nella Tabella 1.4, basata su una presente in X.800, sono indicate le relazioni tra i servizi e i meccanismi di sicurezza.

1.6 Un modello per la sicurezza di rete

Nella Figura 1.5 sono rappresentati schematicamente molti degli argomenti che verranno trattati. Un messaggio deve essere trasferito da una parte a un'altra, attraverso un certo tipo di internet. Le due parti, che costituiscono i **principal** in questa transazione, devono cooperare affinché lo scambio possa avere luogo. Viene stabilito un canale logico di informazione mediante la definizione di un percorso attraverso l'internet dalla sorgente alla destinazione e mediante l'uso cooperativo di protocolli di comunicazione (come TCP/IP) da parte dei due principal.

Gli aspetti di sicurezza entrano in gioco quando è necessario o desiderabile proteggere la trasmissione delle informazioni da un avversario che potrebbe rappresentare una minaccia a riservatezza, autenticità e così via. Tutte le tecniche per fornire sicurezza hanno due componenti.

- Una trasformazione di sicurezza sull'informazione da inviare. Esempi comprendono la cifratura del messaggio, che rende il messaggio illeggibile all'avversario, e l'aggiunta di

Servizi	Meccanismi						
	Cifratura	Firma digitale	Controllo degli accessi	Integrità dei dati	Scambio di autenticazioni	Traffico di riempimento	Controllo dell'indirizzamento
Autenticazione dell'entità partaria	>	>	>	>	>	>	>
Autenticazione dell'origine dei dati	>	>	>	>	>	>	>
Controllo degli accessi			>				
Riservatezza					>	>	>
Riservatezza del flusso di traffico					>		
Integrità dei dati						>	
Non ripudio							
Disponibilità							

Tabella 1.4 Relazione tra servizi e meccanismi di sicurezza.

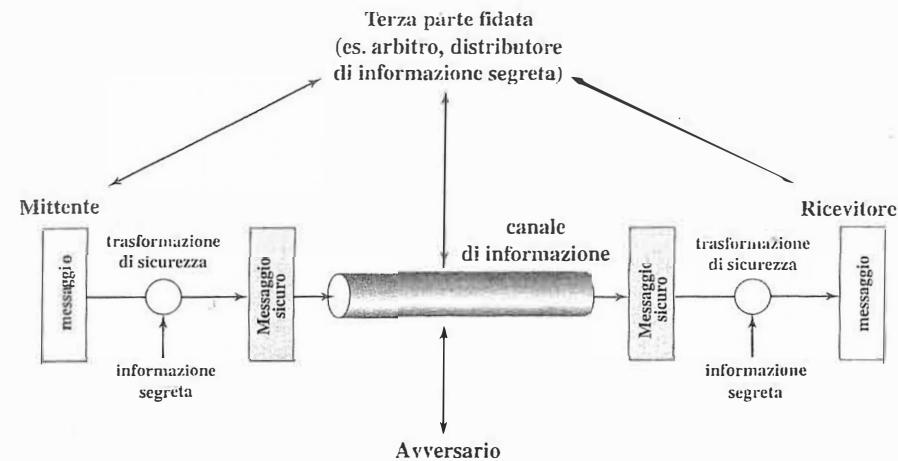


Figura 1.5 Schema per la sicurezza di rete.

un codice basato sul contenuto del messaggio, che può essere utilizzato per verificare l'identità del mittente.

- Qualche informazione segreta condivisa dai due principali e sconosciuta, si spera, all'avversario. Un esempio è la chiave di cifratura utilizzata in combinazione con la trasformazione per effettuare la cifratura del messaggio prima della trasmissione ed effettuare l'inverso (decifratura) alla ricezione⁴.

Per ottenere una trasmissione sicura può essere necessaria una terza parte fidata. Ad esempio, una terza parte può essere responsabile della distribuzione ai due principali delle informazioni segrete mantenendole, nel contempo, nascoste a qualunque avversario. Una terza parte potrebbe essere necessaria per dirimere dispute fra i due principali relativamente all'autenticità della trasmissione di un messaggio.

Questo schema generale mostra i quattro compiti basilari nella progettazione di uno specifico servizio di sicurezza.

1. Progettazione di un algoritmo per eseguire la trasformazione di sicurezza. L'algoritmo deve essere progettato in modo che un avversario non possa farne fallire la funzionalità.
2. Generazione delle informazioni segrete da utilizzare nell'algoritmo.
3. Sviluppo di metodi per la distribuzione e la condivisione delle informazioni segrete.
4. Specifica di un protocollo per i due principali, che utilizza l'algoritmo di sicurezza e le informazioni segrete per ottenere uno specifico servizio di sicurezza.

⁴ Il Capitolo 3 tratta una tipologia di cifratura, conosciuta come cifratura a chiave pubblica, in cui solo uno dei due principali deve possedere informazioni segrete.

La seconda parte di questo libro si concentra sulle tipologie di meccanismi di sicurezza e sui servizi che si inseriscono nello schema della Figura 1.5. Esistono comunque altri interessanti aspetti connessi con la sicurezza che, seppur non contemplati in questo schema, sono tuttavia trattati nel libro. Nella Figura 1.6 è mostrato un modello generale, relativo alla protezione di un sistema informativo da accessi non desiderati, valido per questi ulteriori aspetti. Molti tra i nostri lettori conoscono le problematiche derivanti dalla presenza di hacker che tentano di penetrare i sistemi accessibili su una rete. Un hacker può semplicemente essere una persona che, senza intenti malevoli, prova soddisfazione a violare e inserirsi in un sistema di elaborazione. In altri casi, l'intruso può essere un impiegato scontento che vuole arrecare danni, o un criminale che cerca di sfruttare le risorse di elaborazione a scopo di lucro (ad esempio, ottenere numeri di carte di credito oppure eseguire trasferimenti illegali di denaro).

Un altro tipo di accesso indesiderato consiste nel collocare una logica nel sistema di elaborazione che ne sfrutta le vulnerabilità e può intaccare sia programmi applicativi sia programmi di utilità generale, quali editor e compilatori. I programmi possono presentare due tipologie di minacce.

- **Minacce di accesso all'informazione**, che intercettano o modificano i dati per conto di utenti che non dovrebbero avere accesso a tali dati.
- **Minacce di servizio**, che sfruttano difetti di un servizio nei sistemi di elaborazione per inibirne l'uso ai legittimi utenti.

Virus e worm sono due esempi di attacchi software che possono essere introdotti in un sistema sia attraverso un dischetto – contenente un programma software di utilità con al suo interno la logica non desiderata – sia attraverso la rete. (Quest'ultimo meccanismo è legato maggiormente alla sicurezza di rete.)

I meccanismi di sicurezza necessari per far fronte ad accessi indesiderati ricadono in due grandi categorie (vedi Figura 1.6). La prima categoria può essere denominata funzione "barriera". Comprende procedure di login basate su password, progettate per negare l'accesso a tutti tranne agli utenti autorizzati, e una logica di protezione progettata per rilevare e rifiutare worm, virus ed altri attacchi di questo tipo. Una volta che un utente o un programma indesiderato ha acquisito l'accesso, la seconda linea di difesa comprende un insieme di controlli interni che monitorano l'attività e analizzano le informazioni memorizzate nel tentativo di scoprire la presenza di intrusi non desiderati. Queste problematiche sono trattate nella terza parte del libro.

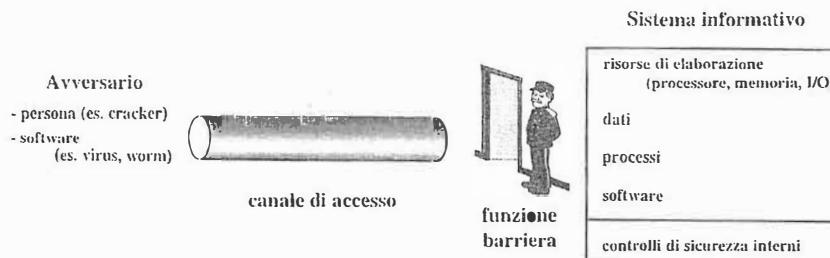


Figura 1.6 Schema di sicurezza per l'accesso alla rete.

1.7 Standard Internet e Internet Society

Molti dei protocolli che costituiscono l'insieme dei protocolli TCP/IP sono stati standardizzati o è in corso il loro processo di standardizzazione. Per convenzione generale, un'organizzazione nota come Internet Society è responsabile dello sviluppo e della pubblicazione di questi standard. L'Internet Society è un'organizzazione che ha come soci professionisti del settore, e che dirige un certo numero di comitati e unità operative coinvolte nello sviluppo e standardizzazione di Internet.

Questo paragrafo fornisce una breve descrizione del modo con cui vengono creati gli standard della famiglia dei protocolli TCP/IP.

Le organizzazioni di Internet e la pubblicazione degli RFC

Internet Society è il comitato coordinatore per la progettazione, ingegnerizzazione e gestione di Internet. Le aree coperte da tale organizzazione comprendono il funzionamento di Internet e la standardizzazione dei protocolli utilizzati dai sistemi connessi a Internet per scopi di interoperabilità. Nell'ambito di Internet Society, esistono tre organizzazioni responsabili dello sviluppo e della pubblicazione di standard.

- **Internet Architecture Board (IAB)**, responsabile della definizione dell'architettura complessiva di Internet, fornisce una guida e una direzione di massima all'IETF.
- **Internet Engineering Task Force (IETF)**, responsabile dell'ingegnerizzazione e dello sviluppo dei protocolli.
- **Internet Engineering Steering Group (IESG)**, responsabile della gestione tecnica delle attività di IETF e del processo di standardizzazione.

I gruppi di lavoro, di cui IETF è costituita, portano avanti l'attuale sviluppo di nuovi standard e protocolli per Internet. L'appartenenza a un gruppo di lavoro è volontaria; chiunque sia interessato può partecipare. Nello sviluppo di una specifica, un gruppo di lavoro prepara una versione bozza del documento disponibile come documento Internet Draft, posizionato nella directory in linea "Internet Drafts" di IETF. Il documento può rimanere nella versione Internet Draft al massimo per sei mesi, e le parti interessate possono revisionarlo e apporare commenti. Durante questo periodo, IESG può approvare la pubblicazione della bozza come documento RFC (*request for comments*). Se la bozza non dovesse progredire allo stato di RFC entro il periodo di sei mesi, viene tolta dalla directory. Successivamente, il gruppo di lavoro può pubblicarne una versione riveduta e corretta.

IETF è responsabile della pubblicazione dei documenti RFC, con approvazione di IESG. RFC sono le annotazioni di lavoro della comunità di ricerca e sviluppo di Internet. Un documento in questa serie può riguardare qualsiasi argomento concernente la comunicazione tra computer e può essere di svariate tipologie, dal resoconto di riunione alla specifica di uno standard.

Il lavoro di IETF si suddivide in otto aree, ciascuna delle quali ha un direttore ed è composta da numerosi gruppi di lavoro. Nella Tabella 1.5 sono mostrate le aree di IETF e i rispettivi campi di interesse.

Area di IETF	Tema	Esempio di gruppi di lavoro
Generale	Processi e procedure IETF	Processi e procedure iETF Infrastruttura delle politiche Processi di organizzazione degli standard Internet
Applicazioni	Applicazioni per Internet	Protocolli legati al Web (HTTP) Integrazione EDI-Internet LDAP
Internet	Infrastruttura di Internet	Ipv6 Estensioni di PPP
Conduzione e gestione	Standard e definizioni per la conduzione della rete	SNMPv3 Remote Network Monitoring
Instrandamento	Protocolli e metodi di gestione per le informazioni di instradamento	Instrandamento multicast OSPF Instrandamento con QoS
Sicurezza	Protocolli e tecnologie per la sicurezza	Kerberos IPsec X.509 S/MIME TLS
Trasporto	Protocolli di livello di trasporto	Servizi differenziati Telefonia IP NFS RSVP
Servizi utente	Metodi per aumentare la qualità delle informazioni disponibili agli utenti di Internet	Uso responsabile di Internet Servizi Utente Documenti FYI

Tabella 1.5 Aree di IETF.

Processo di standardizzazione

La decisione di quali documenti RFC diventeranno standard Internet è presa da IESG, su proposta di IETF. Per diventare uno standard, una specifica deve rispondere ai seguenti criteri:

- essere stabile e ben compresa;
- essere tecnicamente pertinente;
- avere molteplici implementazioni, indipendenti e interoperabili, con effettiva sperimentazione;
- godere di un consenso pubblico significativo;
- essere riconoscibilmente utile per alcune o tutte le parti di Internet.

La differenza basilare fra questi criteri e quelli utilizzati da ISO e ITU-T per gli standard internazionali è costituita dall'enfasi posta sulla sperimentazione operativa.

La parte sinistra della Figura 1.7 mostra la serie di passi, denominata **iter degli standard (standards track)**, che una specifica deve compiere per diventare uno standard; questo processo è definito nel documento RFC 2026. I passi coinvolgono quantitativi crescenti di esami critici e test. A ogni passo, IETF deve inoltrare una proposta per l'avanzamento del protocollo e IESG deve ratificarla. Il processo inizia quando IESG approva la pubblicazione di un documento Internet Draft come un documento RFC con lo status di Proposed standard (standard proposto).

I rettangoli bianchi nel diagramma della Figura 1.7 rappresentano stati transitori in cui un documento deve rimanere per un tempo ragionevolmente breve, e comunque non inferiore al minimo richiesto. Così, un documento deve rimanere nello stato di Proposed standard per almeno sei mesi e in quello di Draft standard (bozza standard) per almeno quattro mesi, tempo necessario per revisioni e commenti. I rettangoli grigi rappresentano stati a lungo termine, che possono essere occupati per anni.

Affinché una specifica assuma lo stato di Draft Standard, devono esserci almeno due sue implementazioni indipendenti e interoperabili dalle quali sia stata maturata un'adeguata esperienza operativa.

Dopo aver ottenuto significative implementazioni ed esperienze operative, una specifica può essere elevata a Internet Standard. A questo punto, alla specifica vengono assegnati un numero STD e un numero RFC. Infine, quando un protocollo diventa obsoleto, viene posto nello stato di Historic (storico).

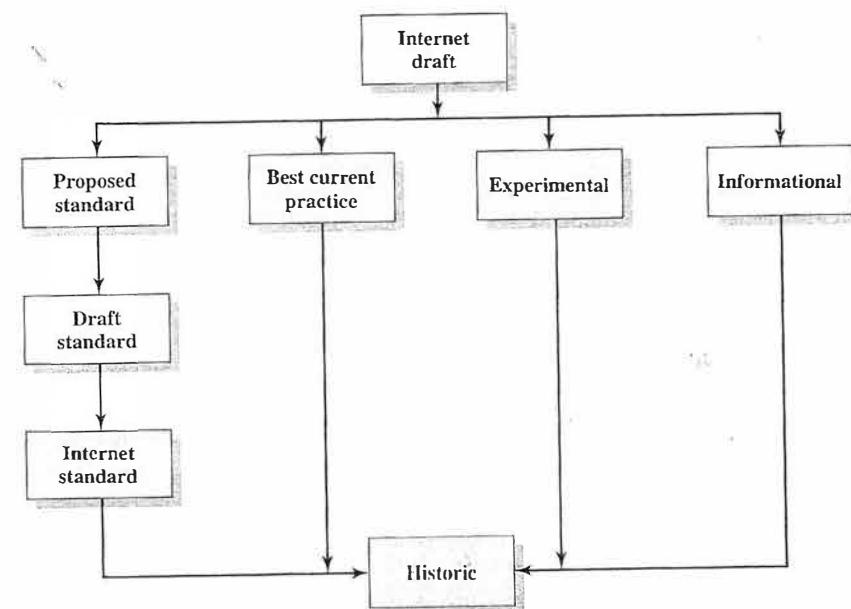


Figura 1.7 Processo di pubblicazione dei documenti Internet RFC.

Categorie degli standard Internet

Tutti gli standard Internet ricadono in due categorie:

- **Specifiche tecniche (TS, technical specification).** Una TS definisce un protocollo, un servizio, una procedura, una convenzione o un formato. La maggior parte degli standard Internet sono TS.
- **Dichiarazioni di applicabilità (AS, applicability statement).** Una AS specifica come e in quali circostanze una o più TS possono essere applicate per sostenere una particolare capacità di Internet. Una AS identifica una o più TS, che sono attinenti alla capacità di Internet, e può specificare valori o intervalli per un particolare parametro associato con una TS o con un sottoinsieme di funzionalità di una TS, che sono attinenti a quella capacità.

Altri tipi di RFC

Esistono numerose RFC che non sono destinate a diventare standard Internet. Alcune RFC standardizzano i risultati dei dibattiti della comunità su dichiarazioni di principi o conclusioni su quale sia il modo migliore per svolgere qualche operazione o funzione di processo IETF. Questo tipo di RFC sono indicate come **Best Current Practice** (BCP). L'approvazione delle BCP segue essenzialmente lo stesso processo di approvazione dei Proposed Standard. Diversamente dai documenti degli standard, non esiste un processo a tre stadi per le BCP: una BCP passa dallo stato di Internet Draft a BCP approvata in un solo passo.

Un protocollo o un'altra specifica non considerata pronta per la standardizzazione può essere pubblicata come un documento **RFC Sperimentale** (*experimental RFC*). Dopo ulteriori raffinamenti, la specifica può essere risottoposta all'iter. Se la specifica è generalmente stabile, ha risolto scelte progettuali note, si ritiene sia ben compresa, ha ricevuto revisioni significative e sembra godere di sufficiente interesse da parte della comunità per essere considerata di valore, allora il documento RFC viene designato a Standard proposto. Infine, vengono pubblicati **documenti di tipo informativo** (*informational specification*) rivolti a fornire notizie generali alla comunità di Internet.

1.8 Contenuti del libro

Questo capitolo svolge la funzione di introduzione. Il resto del libro è organizzato in tre parti.

- **Parte prima.** Fornisce una breve rassegna degli algoritmi crittografici e dei protocolli che stanno al di sotto delle applicazioni di sicurezza di rete, comprese la cifratura, le funzioni di hash, le firme digitali e lo scambio di chiavi.
- **Parte seconda.** Esamina l'uso degli algoritmi crittografici e dei protocolli di sicurezza per fornire sicurezza sulle reti e su Internet. Gli argomenti trattati comprendono l'autenticazione utente, la posta elettronica e la sicurezza IP e Web.
- **Parte terza.** Si occupa delle funzionalità di sicurezza progettate per proteggere un sistema di elaborazione dalle minacce alla sicurezza, che includono intrusi, virus e worm. In questa parte si prenderà in considerazione la tecnologia dei firewall.

Molti degli algoritmi crittografici, dei protocolli e delle applicazioni di sicurezza di rete, descritti in questo libro, sono stati specificati come standard. I più importanti di questi sono gli Standard Internet, definiti nelle RFC (*request for comment*, richieste di commenti) di Internet, e i Federal Information Processing Standard (FIPS), definiti dal National Institute of Standards and Technology (NIST).

1.9 Letture consigliate

[PFLE02] fornisce una buona introduzione alla sicurezza sia dei sistemi di elaborazione sia delle reti. Due altri eccellenti studi sono [PIEP03] e [BISH05]. [BISH03] copre pressappoco gli stessi argomenti di [BISH05], ma con maggiori dettagli e rigore matematico. [SCHIN00] è una lettura preziosa per ogni professionista nel campo della sicurezza dei computer e della rete: tratta le limitazioni della tecnologia, in particolare della crittografia, nel fornire sicurezza, e la necessità di considerare l'hardware, le implementazioni del software, le reti e le persone coinvolte nel provvedere alla sicurezza o nell'attaccarla.

1.10 Internet e risorse Web

Attraverso Internet e la rete web sono disponibili numerose risorse a supporto di questo libro, per mantenersi aggiornati sugli sviluppi in questo campo.

Siti web

Una pagina web speciale è stata preparata per questo libro sul sito WilliamStallings.com/NetSec/NetSec3e.html. Il sito include i seguenti collegamenti.

- **Siti web utili** (*useful web sites*): contiene collegamenti ad altri rilevanti siti web, organizzati per capitoli, compresi i siti elencati in questo paragrafo e nel resto del libro.
- **Errata Corrige**: viene mantenuta e aggiornata, quando necessario, una lista di correzioni degli errori presenti in questo libro. Si prega di comunicare all'autore per posta elettronica qualsiasi errore troviate. Gli errata corrispondenti degli altri libri dell'autore si trovano sul sito WilliamStallings.com.
- **Figure**: tutte le figure di questo libro in formato PDF (Adobe Acrobat).
- **Tabelle**: tutte le tabelle di questo libro in formato PDF.
- **Slide**: un insieme di slide Power Point, organizzate per capitolo.
- **Corsi di sicurezza**: collegamenti a pagine web di corsi basati sul libro; queste pagine possono essere utili ai docenti per la strutturazione dei loro corsi.

Viene anche mantenuto il Computer Science Student Resource Site (pagine di supporto agli studenti di informatica) in WilliamStallings.com/StudentSupport.html. Lo scopo di questo sito è fornire documenti, informazioni e collegamenti a studenti e professionisti nel campo dell'informatica. I collegamenti e i documenti sono organizzati in quattro categorie.

- **Matematica:** comprende matematica di base, manuali elementari sull'analisi delle code e dei sistemi numerici e collegamenti a numerosi siti matematici.
- **How-to:** suggerimenti e guide per risolvere problemi ed esercizi, scrivere rapporti tecnici e preparare presentazioni tecniche.
- **Risorse di ricerca:** collegamenti a importanti raccolte di articoli, relazioni tecniche e bibliografie.
- **Miscellanea:** una varietà di altri documenti e collegamenti utili.

Altri siti web

Esistono numerosi siti web che forniscono informazioni correlate agli argomenti trattati in questo volume. Nel paragrafo *Lettura e siti web consigliati* dei capitoli successivi è possibile trovare riferimenti a specifici siti web. I loro indirizzi (URL) non sono stati inclusi nel testo in quanto tendono a cambiare frequentemente, ma è possibile trovare il collegamento appropriato sul sito web di questo libro.

Altri collegamenti non menzionati in questo testo saranno aggiunti al sito col passare del tempo.

Ecco un elenco di siti web di interesse generale inerenti la crittografia e la sicurezza di rete.

- **COAST:** insieme di collegamenti relativi a crittografia e sicurezza di rete.
- **IETF Security Area:** informazioni aggiornate su iniziative di standardizzazione nell'area della sicurezza di Internet.
- **Computer and Network Security Reference Index:** indice di rivenditori e prodotti commerciali, FAQ, archivi di newsgroup, articoli e altri siti web.
- **The Cryptography FAQ:** voluminoso e utile insieme di FAQ che coprono tutti gli aspetti della crittografia.
- **Tom Dunigan's Security Page:** eccellente elenco di riferimenti a siti web di crittografia e sicurezza di rete.
- **IEEE Technical Committee on Security and Privacy:** copie delle newsletter di IEEE e informazioni su attività relative a IEEE.
- **Computer Security Resource Center:** curato dal National Institute of Standards and Technology (NIST); contiene un'ampia gamma di informazioni sulle minacce, sulla tecnologia e sugli standard della sicurezza.
- **Security Focus:** un'ampia varietà di informazioni sulla sicurezza, con un'enfasi sui prodotti dei fornitori e ciò che riguarda gli utenti finali.
- **SANS Institute:** simile al Security Focus. Esauriente raccolta di articoli.
- **Data Protection Resource Directory:** miscellanea di collegamenti.

Newsgroup USENET

Un certo numero di newsgroup USENET è dedicato alla sicurezza di rete o alla crittografia. Come accade con quasi tutti i gruppi USENET, il rapporto rumore-segnale è elevato, ma vale la pena provare per verificare se qualcuno soddisfa le vostre esigenze. Ecco alcuni tra i più significativi.

- **sci.crypt.research:** il miglior gruppo da seguire. Questo è un newsgroup con moderatore che tratta argomenti di ricerca; i messaggi inseriti devono avere qualche attinenza con gli aspetti tecnici legati alla crittografia.
- **sci.crypt:** discussione generale di crittografia e argomenti correlati.
- **sci.crypt.random-numbers:** gruppo di discussione sulla resistenza crittografica dei numeri casuali.
- **alt.security:** gruppo di discussione generale sugli argomenti di sicurezza.
- **comp.security.misc:** gruppo di discussione generale sugli argomenti di sicurezza informatica.
- **comp.security.firewalls:** discussione generale su prodotti e tecnologie firewall.
- **comp.security.announce:** news e annunci dal CERT.
- **comp.risks:** gruppo di discussione dei rischi per il pubblico, da computer e utenti.
- **comp.virus:** gruppo di discussione con moderatore sui virus dei computer.

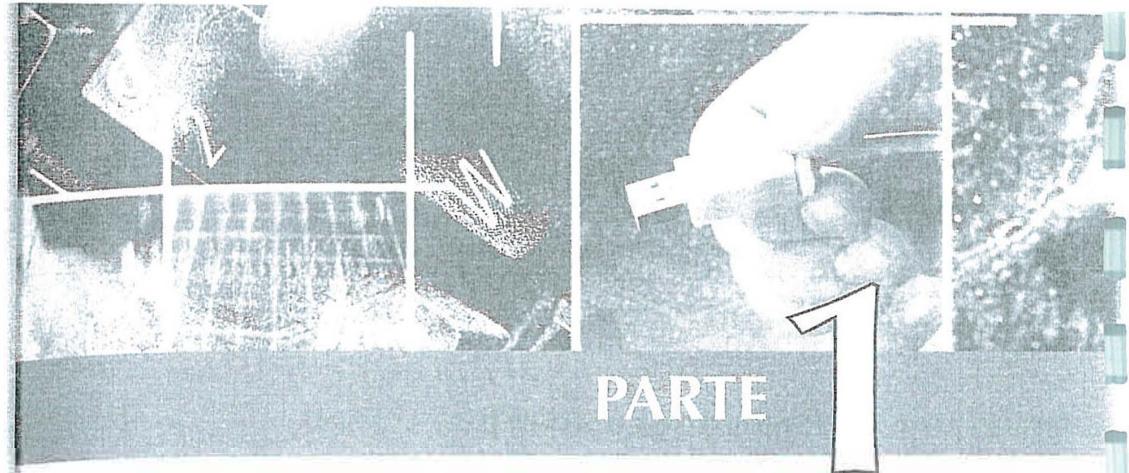
1.11 Domande di revisione ed esercizi

Domande di revisione

- 1.1 Che cos'è l'architettura di sicurezza OSI?
- 1.2 Qual è la differenza tra le minacce di sicurezza attive e passive?
- 1.3 Elenca e definisci brevemente le categorie degli attacchi alla sicurezza attivi e passivi.
- 1.4 Elenca e definisci brevemente le categorie dei servizi di sicurezza.
- 1.5 Elenca e definisci brevemente le categorie dei meccanismi di sicurezza.

Esercizi

- 1.1 Disegna una matrice simile alla Tabella 1.4, che mostri la relazione tra i servizi di sicurezza e gli attacchi.
- 1.2 Disegna una matrice simile alla Tabella 1.4, che mostri la relazione tra i meccanismi di sicurezza e gli attacchi.



Crittografia

La cifratura rappresenta il più rilevante strumento automatizzato per la sicurezza di rete e delle comunicazioni. Due sono le forme di cifratura di utilizzo comune: la cifratura convenzionale, o simmetrica, e la cifratura a chiave pubblica, conosciuta anche come cifratura asimmetrica. La prima parte del libro fornisce una rassegna dei principi alla base della crittografia simmetrica e a chiave pubblica, illustrando gli algoritmi più comunemente usati e affrontando l'applicabilità di base dei due approcci.

Capitolo 2 Crittografia simmetrica e riservatezza dei messaggi

Il Capitolo 2 si focalizza sulla cifratura simmetrica, con particolare enfasi alle tecniche più diffusamente utilizzate: Data Encryption Standard (DES), il suo successore, la versione a tre chiavi di DES, e l'Advanced Encryption Standard (AES). Oltre agli aspetti relativi all'effettiva realizzazione di un algoritmo di cifratura simmetrico, esistono numerose problematiche di progetto che fanno riferimento all'uso della cifratura simmetrica per fornire riservatezza. Il capitolo affronta inoltre l'argomento della cifratura di messaggi di dimensioni elevate, la cifratura a livello mittente-ricevente (*end-to-end encryption*) contrapposta alla cifratura a livello di collegamento (*link encryption*), e le tecniche di distribuzione delle chiavi.

Capitolo 3 Crittografia a chiave pubblica e autenticazione dei messaggi

Come misura di sicurezza, l'autenticazione ricopre un ruolo importante quanto quello della riservatezza. A livello minimo, l'autenticazione assicura la provenienza di un messaggio dalla sorgente dichiarata. Inoltre, l'autenticazione può comprendere la protezione da modifiche, ritardi, attacchi di replay e alterazione della sequenza dei messaggi. Il Capitolo 3 inizia con l'analisi dei requisiti di autenticazione e successivamente illustra i vari approcci. Un elemento chiave di ogni schema di autenticazione è l'utilizzo di un "autenticatore", di solito costituito da un codice di autenticazione di messaggio (MAC, *message authentication code*)

o da una funzione hash. Il capitolo analizza le scelte progettuali per entrambi i tipi di algoritmi e illustra numerosi specifici esempi.

La cripotecnica a chiave pubblica, che costituisce l'altra fondamentale tecnica di cripotecnica in aggiunta a quella convenzionale, ha rivoluzionato la sicurezza delle comunicazioni. Il Capitolo 3 fornisce un'introduzione alla cripotecnica a chiave pubblica, analizzando in dettaglio l'algoritmo RSA e considerando le problematiche relative alla gestione delle chiavi. Inoltre, presenta la tecnica Diffie-Hellman per lo scambio di chiavi e descrive quindi le firme digitali e le loro applicazioni.

Capitolo 2

Cripotecnica simmetrica e riservatezza dei messaggi

La cripotecnica simmetrica – chiamata anche cripotecnica convenzionale, cripotecnica a chiave segreta o cripotecnica a chiave singola – era l'unico tipo di cripotecnica utilizzato prima dello sviluppo della cripotecnica a chiave pubblica avvenuto verso la fine degli anni '70¹. Ed è di gran lunga la più utilizzata tra i due tipi di cripotecnica.

Questo capitolo inizia illustrando un modello di riferimento generale per il processo di cripotecnica simmetrica; tale modello consentirà di comprendere il contesto in cui gli algoritmi sono utilizzati. Successivamente, verranno descritti tre importanti algoritmi di cripotecnica a blocchi: DES, triplo DES e AES. Infine, il capitolo introduce la cripotecnica simmetrica del flusso e descrive RC4, un sistema di cripotecnica del flusso ampiamente utilizzato.

Si illustrerà poi come applicare questi algoritmi per ottenere la riservatezza.

2.1 Elementi di cripotecnica simmetrica

Uno schema di cripotecnica simmetrica è costituito da cinque elementi (Figura 2.1).

- **Testo in chiaro:** è il messaggio o dato originario, fornito all'algoritmo come ingresso.
- **Algoritmo di cripotecnica:** effettua numerose sostituzioni e trasformazioni sul testo in chiaro.
- **Chiave segreta:** è anch'essa fornita in ingresso all'algoritmo. Le specifiche sostituzioni e trasformazioni effettuate dall'algoritmo dipendono dalla chiave.
- **Testo cifrato:** è il messaggio cifrato prodotto in uscita dall'algoritmo. Tale messaggio dipende dal testo in chiaro e dalla chiave segreta. A parità di messaggio, due chiavi diverse daranno origine a due diversi testi cifrati.
- **Algoritmo di decripotecnica:** è essenzialmente l'inverso dell'algoritmo di cripotecnica; riceve in ingresso il testo cifrato e la chiave segreta ricevuta in ingresso dall'algoritmo di cripotecnica, e restituisce il testo originario in chiaro.

¹ La cripotecnica a chiave pubblica è stata descritta per la prima volta in letteratura nel 1976; la National Security Agency (NSA) sostiene però di averla inventata alcuni anni prima.

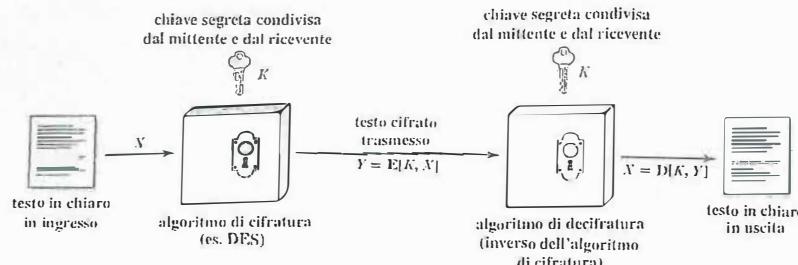


Figura 2.1 Modello semplificato di cifratura simmetrica.

Due requisiti sono alla base di un utilizzo sicuro della cifratura simmetrica.

1. È necessario un algoritmo di cifratura robusto. Come requisito minimo, l'algoritmo deve essere tale che un avversario – a conoscenza dell'algoritmo e con accesso a uno o più testi cifrati – non sia in grado di decifrare il testo cifrato o di calcolare la chiave. Di tale requisito si fornisce di solito una formulazione più restrittiva: l'avversario non deve essere in grado di decifrare il testo cifrato o di scoprire la chiave anche se in possesso di un certo numero di testi cifrati e dei corrispondenti testi in chiaro.
2. Mittente e ricevente devono aver ottenuto copie della chiave segreta in modo sicuro e devono mantenere la chiave in un luogo sicuro. Tutte le comunicazioni effettuate mediante la chiave diventano leggibili se qualcuno a conoscenza dell'algoritmo di cifratura è in grado di scoprire la chiave.

È importante notare che la sicurezza della cifratura simmetrica dipende dalla segretezza della chiave piuttosto che dalla segretezza dell'algoritmo. Si assume cioè che sia impossibile decifrare un messaggio avendo a disposizione il testo cifrato e conoscendo l'algoritmo di cifratura/decifratura. In altri termini, non è necessario mantenere segreto l'algoritmo; è sufficiente assicurare la segretezza della chiave.

Questa caratteristica rende la cifratura simmetrica adatta a un utilizzo su larga scala. Il fatto che non sia necessario mantenere segreto l'algoritmo consente ai produttori di sviluppare, come hanno fatto, implementazioni degli algoritmi di cifratura dei dati su chip a basso costo. Questi chip sono disponibili su vasta scala e incorporati in numerosi prodotti.

Con l'uso della cifratura simmetrica, il principale problema di sicurezza è mantenere segreta la chiave.

Crittografia

I sistemi crittografici sono generalmente classificati in base a tre criteri indipendenti.

1. **Il tipo di operazioni utilizzate per trasformare il testo in chiaro in testo cifrato.** Tutti gli algoritmi di cifratura si basano su due principi generali: la **sostituzione** (*substitution*), con cui ogni elemento del testo in chiaro (sia esso un bit, una lettera dell'alfabeto, o un gruppo di bit o lettere) è trasformato in un altro elemento, e la **trasposizione** (*transposition*), con cui gli elementi del testo in chiaro sono riorganizzati.

Il requisito fondamentale è che non si verifichi perdita di informazioni (cioè, che tutte le operazioni siano reversibili). Molti sistemi, cui si fa riferimento come sistemi di prodotti, comportano più iterazioni del processo di sostituzione e trasposizione.

2. **Il numero di chiavi utilizzate.** Se il mittente e il ricevente utilizzano la stessa chiave, il sistema è chiamato simmetrico, a chiave singola, a chiave segreta o sistema di cifratura convenzionale. Se la chiave utilizzata dal mittente è diversa da quella utilizzata dal ricevente, il sistema è chiamato asimmetrico, a due chiavi o sistema di cifratura a chiave pubblica.
3. **Il modo in cui il testo in chiaro è elaborato.** Un **cifrario a blocchi** (*block cipher*) elabora quanto gli viene fornito in ingresso un blocco alla volta, producendo in uscita un blocco per ogni blocco ricevuto in ingresso. Un **cifrario a flusso** (*stream cipher*) elabora gli elementi in ingresso in maniera continua, producendo come uscita un elemento alla volta, man mano che l'elaborazione procede.

Crittoanalisi

La crittoanalisi è il processo mediante il quale si tenta di risalire al testo in chiaro o alla chiave. La strategia impiegata dal crittoanalista dipende dal tipo di sistema di cifratura utilizzato e dalle informazioni in suo possesso.

La Tabella 2.1 riassume le principali tipologie di attacchi di crittoanalisi, in base alla quantità di informazioni in possesso del crittoanalista. Il tipo di attacco più difficile da perpetrare si verifica quando l'unico elemento conosciuto è il testo cifrato. In alcuni casi, non si conosce neppure l'algoritmo di cifratura, ma generalmente si può ipotizzare che l'algoritmo utilizzato per la cifratura sia noto.

In questo caso, un tipo di attacco perpetrabile è l'utilizzo del cosiddetto approccio “a forza bruta”, che consiste nel tentare tutte le chiavi possibili. Se lo spazio delle chiavi è molto ampio, l'attacco diventa impraticabile. In questo caso, l'avversario deve fare affidamento su un'analisi del testo cifrato, analisi che generalmente viene effettuata mediante l'applicazione di test statistici. Per poter utilizzare questo approccio, l'avversario deve avere un'idea di massima di quale sia il tipo di testo in chiaro nascosto, ad esempio se è un testo in inglese o in italiano, un file eseguibile, un codice sorgente Java, un file di contabilità e così via.

L'attacco che si basa sulla conoscenza del solo testo cifrato è il più semplice da cui difendersi in quanto l'avversario è in possesso della quantità minima di informazione su cui operare. In molti casi, però, l'analista è in possesso di una maggiore quantità di informazioni. Ad esempio, può essere in grado di ottenere uno o più messaggi in chiaro insieme alle corrispondenti cifrature. Oppure, può essere a conoscenza del fatto che alcune sequenze che compaiono nel testo in chiaro appariranno anche in un messaggio. Ad esempio, un file codificato in formato Postscript inizia sempre con la stessa sequenza di caratteri, così come un messaggio relativo a un trasferimento elettronico di fondi può avere un'intestazione standard. Tutti questi sono esempi di **testo in chiaro conosciuto** (*known plaintext*). Con questo tipo di conoscenza, l'analista può essere in grado di dedurre la chiave, basandosi sul modo in cui il testo in chiaro conosciuto viene trasformato.

Strettamente correlato all'attacco basato su testo in chiaro conosciuto, è quello che può essere definito attacco basato su **parole probabili**. Se l'avversario sta studiando la cifratura di un generico messaggio di testo, avrà una minima conoscenza di cosa il messaggio rappresenta. Se invece l'avversario è alla ricerca di specifiche informazioni, allora alcune parti

del messaggio possono essere conosciute. Ad esempio, se viene trasmesso un intero file di registrazioni, l'avversario può essere a conoscenza della dislocazione di alcune parole chiave nell'intestazione del file. Oppure, il codice sorgente di un programma sviluppato da una società potrebbe includere, in una posizione standard, una dichiarazione di copyright.

Se l'analista riesce, con qualche mezzo, ad accedere al sistema sorgente e a inserirvi un messaggio da lui scelto, questo rende possibile un attacco mediante **testo in chiaro selezionato** (*choose-plaintext*). Di solito, se l'analista può scegliere il testo da cifrare, può deliberatamente scegliere sequenze che molto probabilmente rivelino la struttura della chiave.

Nella Tabella 2.1 sono elencati altri due tipi di attacchi: l'attacco basato su testo cifrato selezionato e quello basato su testo selezionato. Questi attacchi sono meno frequentemente utilizzati come tecniche di crittoanalisi ma sono comunque vie praticabili per un attacco.

Soltanto algoritmi relativamente poco robusti non riescono a resistere a un attacco basato solo sul testo cifrato. Solitamente, un algoritmo di cifratura è progettato per resistere a un attacco basato su testo in chiaro conosciuto.

Vi sono due ulteriori definizioni degne di nota. Un sistema di cifratura è **computazionalmente sicuro** se il testo cifrato generato soddisfa uno dei seguenti criteri:

- il costo per rendere inefficace il cifrario supera il valore dell'informazione cifrata;
- il tempo richiesto per rendere inefficace il cifrario supera l'arco temporale in cui l'informazione ha una qualche utilità.

Tipo di attacco	Informazioni in possesso del crittoanalista
Solo testo cifrato	<ul style="list-style-type: none"> • Algoritmo di cifratura • Testo cifrato da decifrare
Testo in chiaro conosciuto	<ul style="list-style-type: none"> • Algoritmo di cifratura • Testo cifrato da decifrare • Uno o più testi in chiaro e corrispondenti testi cifrati ottenuti mediante la chiave segreta
Testo in chiaro selezionato	<ul style="list-style-type: none"> • Algoritmo di cifratura • Testo cifrato da decifrare • Testo in chiaro scelto dal crittoanalista e corrispondente testo cifrato generato mediante la chiave segreta
Testo cifrato selezionato	<ul style="list-style-type: none"> • Algoritmo di cifratura • Testo cifrato da decifrare • Testo cifrato, con significato, scelto dal crittoanalista e corrispondente testo in chiaro decifrato mediante l'applicazione della chiave segreta al testo cifrato
Testo selezionato	<ul style="list-style-type: none"> • Algoritmo di cifratura • Testo cifrato da decifrare • Testo in chiaro scelto dal crittoanalista e corrispondente testo cifrato generato mediante la chiave segreta • Testo cifrato con significato scelto dal crittoanalista e corrispondente testo in chiaro decifrato mediante l'applicazione della chiave segreta al testo cifrato

Tabella 2.1 Tipologie di attacchi effettuabili su messaggi cifrati.

Dimensione della chiave (in bit)	Numero di possibili chiavi	Tempo necessario a 1 decifratura/ μ s	Tempo necessario a 10^6 decifrazioni/ μ s
32	$2^{32} = 4,3 \times 10^9$	$2^{31}\mu\text{s} = 35,8 \text{ minuti}$	2,15 millisecondi
56	$2^{56} = 7,2 \times 10^{16}$	$2^{55}\mu\text{s} = 1142 \text{ anni}$	10,01 ore
128	$2^{128} = 3,4 \times 10^{38}$	$2^{127}\mu\text{s} = 5,4 \times 10^{24} \text{ anni}$	$5,4 \times 10^{18} \text{ anni}$
168	$2^{168} = 3,7 \times 10^{50}$	$2^{167}\mu\text{s} = 5,9 \times 10^{36} \text{ anni}$	$5,9 \times 10^{30} \text{ anni}$
26 caratteri (permutazione)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\mu\text{s} = 6,4 \times 10^{12} \text{ anni}$	$6,4 \times 10^6 \text{ anni}$

Tabella 2.2 Tempo medio richiesto per una ricerca esaustiva delle chiavi.

Sfortunatamente, stimare lo sforzo richiesto per effettuare con successo la crittoanalisi del testo cifrato è molto difficile. L'approccio a forza bruta è indicato quando si può ipotizzare che nell'algoritmo non siano presenti debolezze intrinseche di tipo matematico. In questo caso si possono effettuare stime ragionevoli su tempi e costi.

Un approccio a forza bruta implica che si debbano provare tutte le chiavi possibili fino a quando si ottiene una trasformazione comprensibile del testo cifrato in testo in chiaro. In media, si devono provare metà di tutte le possibili chiavi per ottenere tale traduzione. La Tabella 2.2 mostra il tempo richiesto in funzione della dimensione delle chiavi. Con l'**algoritmo DES** (*data encryption standard*) sono utilizzate chiavi di dimensione 56 bit. Per ogni dimensione di chiave considerata, i risultati sono illustrati assumendo che ciascuna singola decifratura richieda 1 microsecondo, in quanto tale ipotesi è ragionevole per gli odierni elaboratori. Con l'utilizzo di architetture di microprocessori altamente parallele, è possibile ottenere tassi di elaborazione di diversi ordini di grandezza più elevati. Nell'ultima colonna della Tabella 2.2 sono considerati i risultati per un sistema in grado di elaborare 1 milione di chiavi al microsecondo. Come si può osservare, con questo tipo di prestazioni l'algoritmo DES non può più essere considerato computazionalmente sicuro.

Struttura del cifrario di Feistel

Molti algoritmi di cifratura simmetrica a blocchi, DES incluso, hanno una struttura che è stata descritta per la prima volta da Horst Feistel di IBM nel 1973 [FEIS73], illustrata nella Figura 2.2. Gli ingressi dell'algoritmo di cifratura sono un blocco di testo in chiaro di lunghezza $2w$ bit e una chiave K . Il blocco di testo in chiaro è diviso in due parti uguali, denotate con L_0 e R_0 . Le due parti vengono sottoposte a n iterazioni e poi combinate per formare il blocco di testo cifrato. La i -esima iterazione riceve in ingresso L_{i-1} e R_{i-1} , risultanti dall'iterazione precedente, e una sottochiave K_i , derivata dalla chiave completa K . Solitamente, le sottochiavi K_i sono diverse l'una dall'altra e differiscono inoltre anche da K e sono generate da quest'ultima tramite un algoritmo di generazione di sottochiavi.

Tutte le iterazioni hanno la stessa struttura. Viene effettuata una sostituzione della metà sinistra del dato in ingresso. Tale sostituzione è effettuata mediante l'applicazione di una funzione F alla metà destra dei dati in ingresso ed effettuando l'**OR esclusivo** (XOR) tra il risultato della funzione e la metà sinistra dei dati. La funzione F ha la stessa struttura per ogni iterazione ma è parametrica rispetto alla sottochiave K_i . Dopo aver effettuato questa sostituzione, viene compiuta una trasposizione che consiste nello scambio delle due metà dei dati.

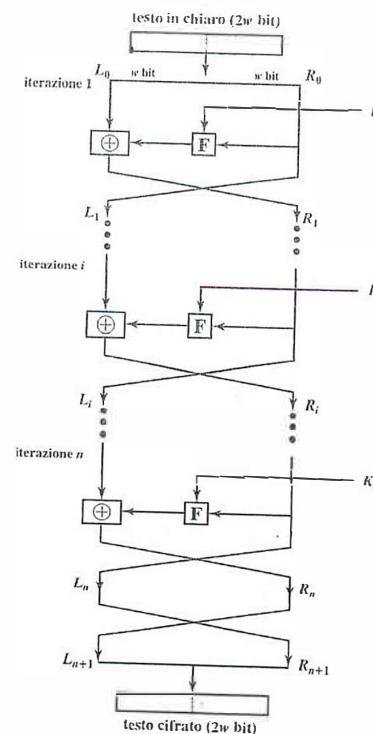


Figura 2.2 Schema di Feistel classico.

La struttura di Feistel è un esempio particolare di una struttura più generale utilizzata da tutti i cifrari simmetrici a blocchi. In generale, un cifrario simmetrico a blocchi consiste in una sequenza di cicli, in ciascuno dei quali si effettua una sostituzione e una trasposizione condizionata dal valore di una chiave segreta. L'esatta realizzazione di cifrario simmetrico a blocchi dipende dalla scelta dei seguenti parametri e dalle seguenti caratteristiche di progetto.

- **Dimensione del blocco.** Dimensioni maggiori del blocco implicano una maggiore sicurezza (a parità di tutti gli altri fattori) ma comportano una ridotta velocità di cifratura/decifratura. Una dimensione del blocco di 128 bit è un ragionevole compromesso, quasi universalmente adottato, nella progettazione di cifrari a blocchi.
- **Dimensione della chiave.** Dimensioni maggiori per la chiave implicano una maggiore sicurezza ma diminuiscono la velocità di cifratura/decifratura. La dimensione della chiave più comunemente utilizzata dagli attuali algoritmi è 128 bit.
- **Numero di iterazioni.** Il principio alla base del cifrario di Feistel è che un'unica iterazione non offre un'adeguata sicurezza che aumenta, però, all'aumentare del numero di iterazioni. Comunemente il numero di iterazioni è pari a 16.

- **Algoritmo per la generazione delle sottochiavi.** Più elevata è la complessità di questo algoritmo più ardua è la crittoanalisi.
- **Funzione di cifratura.** Anche in questo caso, una complessità elevata significa di solito una maggiore resistenza alla crittoanalisi.

Nel progetto di un cifrario simmetrico a blocchi occorre tener presente due ulteriori considerazioni.

- **Cifratura/decifratura software veloce.** In molte circostanze, la cifratura è incorporata in applicazioni o funzioni di utilità e questo preclude una sua implementazione hardware. Di conseguenza, la velocità di esecuzione dell'algoritmo diventa di primaria importanza.
- **Semplicità di analisi.** Sebbene sia desiderabile rendere l'algoritmo di cifratura il più possibile difficile da crittoanalizzare, è vantaggioso rendere l'algoritmo semplice da analizzare. Questo perché, se l'algoritmo può essere descritto in modo chiaro e conciso, è più semplice analizzarlo rispetto alla sua vulnerabilità alla crittoanalisi e, quindi, fornire un più elevato livello di certificazione della sua robustezza. Ad esempio, l'algoritmo DES non è semplice da analizzare.

La decifratura mediante un cifrario simmetrico a blocchi è praticamente analoga alla cifratura. La regola da seguire è la seguente: si utilizza il testo cifrato come ingresso dell'algoritmo, utilizzando le sottochiavi K_i in ordine inverso rispetto alla cifratura. Cioè, si usa K_n alla prima iterazione, K_{n-1} alla seconda, e così via fino a utilizzare K_1 nell'ultima iterazione. La modalità con cui viene effettuata la decifratura è un'apprezzabile caratteristica del cifrario di Feistel, in quanto non richiede di implementare due algoritmi differenti per cifratura e decifratura.

2.2 Algoritmi di cifratura simmetrica a blocchi

Gli algoritmi di cifratura simmetrica più utilizzati sono i cifrari a blocchi. Un cifrario a blocchi elabora il testo in chiaro in ingresso in blocchi di dimensione fissa e genera, per ogni blocco di testo in chiaro, un blocco di testo cifrato di ugual dimensione. Questo paragrafo si focalizza sui tre cifrari simmetrici a blocchi: DES, 3DES e AES.

Data Encryption Standard

Il sistema di cifratura più utilizzato è quello definito nell'ambito del Data encryption standard (DES) adottato nel 1977 dal National Bureau of Standards – oggi National Institute of Standards and Technology (NIST) – come Federal Information Processing Standard 46 (FIPS PUB 46). L'algoritmo su cui è basato DES è chiamato DEA (*data encryption algorithm*)².

² La terminologia utilizzata è leggermente imprecisa. Fino a poco tempo fa, i termini DES e DEA avrebbero potuto essere utilizzati come sinonimi. Tuttavia, la versione più recente di DES contiene una specifica di DEA, qui descritta, oltre che del triplo DEA (3DES), descritto successivamente. Sia DEA che 3DES sono parte del Data Encryption Standard. Inoltre, fino alla recente adozione del termine ufficiale 3DES, l'algoritmo del triplo DES era comunemente chiamato triplo DES e scritto come 3DES. Per comodità si userà 3DES.

Descrizione dell'algoritmo

La dimensione del testo in chiaro è di 64 bit mentre la dimensione della chiave è di 56 bit; testi in chiaro di dimensioni maggiori vengono elaborati a blocchi di 64 bit. La struttura DES è una piccola variazione della rete di Feistel mostrata nella Figura 2.2. Ci sono 16 iterazioni. Dalla chiave originale vengono generate 16 sottochiavi, ciascuna delle quali viene usata in una iterazione.

Il processo di decifratura in DES è essenzialmente analogo al processo di cifratura. Il procedimento è il seguente: il testo cifrato viene utilizzato come ingresso dell'algoritmo DES, ma le chiavi K_i sono utilizzate in ordine inverso rispetto a quello della cifratura. Cioè, si usa la chiave K_{16} nella prima iterazione, K_{15} nella seconda, e così via finché K_1 non viene utilizzata nella sedicesima e ultima iterazione.

Robustezza di DES

Le perplessità circa la robustezza di DES sono essenzialmente di due tipi: quelle riguardanti l'algoritmo stesso e quelle riguardanti l'utilizzo di una chiave di 56 bit. Un primo aspetto riguarda la possibilità di effettuare crittoanalisi sfruttando le caratteristiche dell'algoritmo DES. Nel corso degli anni, sono stati effettuati molti tentativi volti a trovare e sfruttare possibili punti deboli nell'algoritmo, e questo ha reso DES l'algoritmo di cifratura più studiato. Nonostante i numerosi approcci, nessuno è finora riuscito a scoprire un punto debole fatale in DES³.

La lunghezza della chiave è l'aspetto più critico. Con una lunghezza della chiave di 56 bit, esistono 2^{56} possibili chiavi, che corrisponde approssimativamente a $7,2 \times 10^{16}$ chiavi. Quindi, a prima vista, un attacco a forza bruta sembra impraticabile. Assumendo che, in media, si deve cercare la metà dello spazio delle chiavi, una singola macchina che esegue una cifratura DES per microsecondo impiegherebbe più di un migliaio d'anni (Tabella 2.2) per violare il cifrario.

Tuttavia, l'assunzione di una cifratura per microsecondo è eccessivamente conservativa. Si dimostrò definitivamente che DES era insicuro nel luglio del 1998, quando la Electronic Frontier Foundation (EFF) annunciò di aver reso inefficace una cifratura DES utilizzando un elaboratore *ad hoc*, detto "DES cracker", costato meno di 250 mila dollari.

Per perpetrare l'attacco furono necessari meno di tre giorni. EFF pubblicò una dettagliata descrizione dell'elaboratore utilizzato, rendendo così possibile ad altri la costruzione del loro programma cracker [EFF98]. Naturalmente, la costante diminuzione dei prezzi dell'hardware e la crescita della velocità dei processori rende oggi DES ancora più vulnerabile ad attacchi.

È importante sottolineare che perpetrare un attacco a forza bruta richiede qualcosa di più che provare tutte le chiavi possibili. Infatti, a meno di fornire un testo in chiaro conosciuto, l'analista deve essere in grado di riconoscere il testo in chiaro come tale. Questo compito è agevole se il messaggio è costituito da testo in chiaro in inglese, sebbene sia necessario automatizzare il procedimento di riconoscimento dell'inglese. Risulta invece più difficile se il messaggio è stato compresso prima della cifratura. Il procedimento diventa ancora più difficile da automatizzare se il messaggio compresso contiene tipi di dati più generali, ad esem-

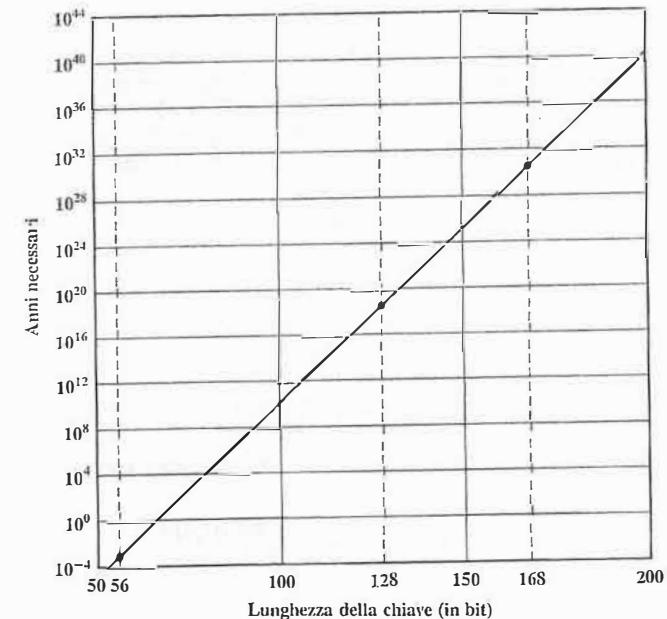


Figura 2.3 Tempo necessario per rendere inefficace un codice (ipotizzando 10^6 decifrature/ μ s).

pio è un file numerico. L'approccio a forza bruta va quindi compendiato con un certo grado di conoscenza del testo in chiaro e con metodi per distinguere automaticamente il testo in chiaro da quello cifrato. L'approccio utilizzato da EFF prende in considerazione entrambi questi aspetti e fornisce alcune tecniche automatizzate, efficaci in differenti contesti.

Una considerazione finale. Se l'unico attacco possibile a un algoritmo di cifratura è quello a forza bruta, allora diventa ovvio il metodo per contrastarlo: utilizzare chiavi più lunghe. Per avere un'idea della dimensione di chiave necessaria, si può utilizzare il cracker sviluppato da EFF come base per effettuare le stime. Il cracker di EFF era un prototipo e si può ipotizzare che con l'odierna tecnologia sia realizzabile, a costi sostenibili, una macchina più veloce. Ipotizzando che un cracker possa effettuare un milione di decifrature al microsecondo, che è il tasso utilizzato nella Tabella 2.2, allora sono necessarie circa 10 ore per rendere inefficace un codice DES. Questo risultato costituisce un'accelerazione di circa un fattore 7 rispetto ai risultati ottenuti da EFF. La Figura 2.3 mostra il tempo necessario – utilizzando questo tasso – per rendere inefficace un algoritmo basato su DES, in funzione della dimensione della chiave. Ad esempio, utilizzando una chiave di 128 bit, che è la dimensione maggiormente utilizzata dai correnti algoritmi, sono necessari più di 10^{18} anni per rendere inefficace il codice utilizzando il cracker di EFF. Anche riuscendo a velocizzare il cracker di un fattore pari a 1 trilione (10^{12}), sarebbero ancora necessari più di 1 milione di anni per rendere inefficace il codice. Di conseguenza, l'utilizzo di una chiave a 128 bit garantisce che l'algoritmo non sia attaccabile tramite l'approccio a forza bruta.

³ O almeno nessuno ha pubblicamente riconosciuto tale scoperta.

Triplo DES

Il triplo DES (3DES) inizialmente fu standardizzato per essere utilizzato in applicazioni finanziarie nello standard ANSI X9.17 nel 1985. 3DES fu incorporato come una parte del Data Encryption Standard nel 1999, con la pubblicazione su FIPS PUB 46-3.

3DES utilizza tre chiavi e tre esecuzioni dell'algoritmo DES. L'algoritmo (Figura 2.4a) compie una sequenza di cifratura-decifratura-cifratura (EDE, *encryption-decryption-encryption*):

$$C = E(K_3, D(K_2, E(K_1, P)))$$

dove

C = testo cifrato

P = testo in chiaro

$E[K, X]$ = cifratura di X mediante la chiave K

$D[K, X]$ = decifratura di X mediante la chiave K

La decifratura consiste nelle stesse operazioni della cifratura, utilizzando però le chiavi in ordine inverso (Figura 2.4b):

$$P = D(K_1, E(K_2, D(K_3, C)))$$

L'utilizzo della decifratura come secondo passo della cifratura 3DES non ha alcun significato dal punto di vista crittografico. L'unico vantaggio è che consente agli utilizzatori di 3DES di decifrare dati cifrati tramite il vecchio DES:

$$C = E(K_1, D(K_1, E(K_1, P))) = E[K, P]$$

Dal momento che 3DES utilizza tre chiavi diverse, è come se avesse una lunghezza effettiva di chiave di 168 bit. FIPS 46-3 consente anche l'utilizzo di due sole chiavi, assumendo $K_1 = K_3$; con questa variante la lunghezza effettiva della chiave è 112 bit. FIPS 46-3 fornisce le seguenti linee guida per quanto riguarda 3DES.

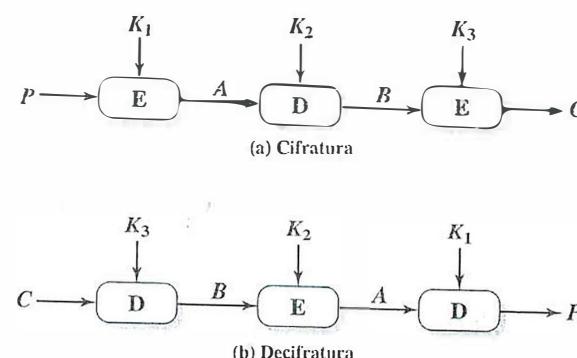


Figura 2.4 Triplo DES.

- 3DES è l'algoritmo di cifratura simmetrica approvato e raccomandato dal FIPS.
- Il DES originario, che utilizza una sola chiave di 56 bit, è permesso nel contesto dello standard solo per sistemi legacy. I nuovi sistemi dovrebbero adottare 3DES.
- Le organizzazioni governative in cui sono presenti sistemi legacy che utilizzano DES sono incoraggiate ad effettuare una transizione a 3DES.
- È previsto che 3DES e l'Advanced Encryption Standard (AES) coesisteranno come algoritmi entrambi approvati dal FIPS, consentendo quindi una transizione graduale verso AES.

È facile convincersi dei pregi dell'algoritmo 3DES, in quanto può vantare la stessa resistenza di DES alla crittoanalisi, dal momento che l'algoritmo crittografico che utilizza è DEA. Inoltre, con una chiave di 168 bit gli attacchi basati sull'approccio a forza bruta sono praticamente irrealizzabili.

Alla fine, AES tenderà a sostituire 3DES, ma questo processo richiederà parecchi anni. NIST ha anticipato che 3DES rimarrà un algoritmo approvato (per l'utilizzo nell'amministrazione U.S.) nell'immediato futuro.

Advanced Encryption Standard

3DES ha due attrattive che gli assicurano l'uso diffuso per i prossimi anni. In primo luogo, con la sua chiave lunga 168 bit, supera la vulnerabilità agli attacchi a forza bruta di DEA. In secondo luogo, l'algoritmo di cifratura utilizzato in 3DES è lo stesso usato in DEA.

Questo algoritmo è stato sottoposto a un numero maggiore di analisi e per un periodo di tempo più ampio rispetto a qualsiasi altro algoritmo di cifratura, e non si è trovato nessun efficace attacco di crittoanalisi ad eccezione dell'approccio a forza bruta. Di conseguenza, esiste un elevato grado di fiducia sulla resistenza di 3DES alla crittoanalisi. Se l'unico fattore da considerare fosse la sicurezza, allora 3DES sarebbe la scelta appropriata come algoritmo standard di cifratura per i prossimi anni.

Il principale svantaggio di 3DES è individuabile nella lentezza dell'implementazione software dell'algoritmo. In effetti, il DEA originario fu progettato per l'hardware della metà degli anni '70 e quindi non porta a realizzazioni software efficienti. 3DES, che compie un numero di iterazioni triplo rispetto a quelle di DES, è di conseguenza ancora più lento. Un'ulteriore limite consiste nel fatto che sia DEA che 3DES utilizzano blocchi di 64 bit. Per ragioni legate sia alla sicurezza che alla efficienza è preferibile utilizzare blocchi di dimensioni maggiori.

I limiti illustrati in precedenza non rendono 3DES un buon candidato per un utilizzo a lungo termine. Per trovare un sostituto, nel 1997 NIST bandì un concorso per la definizione di un nuovo Advanced Encryption Standard (AES), che possedesse una robustezza uguale o migliore rispetto a 3DES, e presentasse miglioramenti significativi in termini di efficienza. In aggiunta a questi requisiti generali, NIST specificò che AES doveva essere un cifrario a blocchi simmetrico, con una lunghezza di blocco pari a 128 bit, utilizzabile con chiavi di lunghezza 128, 192, e 256 bit. I criteri di valutazione delle proposte comprendono la sicurezza, l'efficienza computazionale, lo spazio di memoria richiesto, l'idoneità per hardware e software, e la flessibilità.

Durante la prima tornata di valutazione, furono accettati 15 algoritmi fra quelli proposti. Durante la seconda tornata la scelta fu ridotta a 5. NIST completò il suo processo

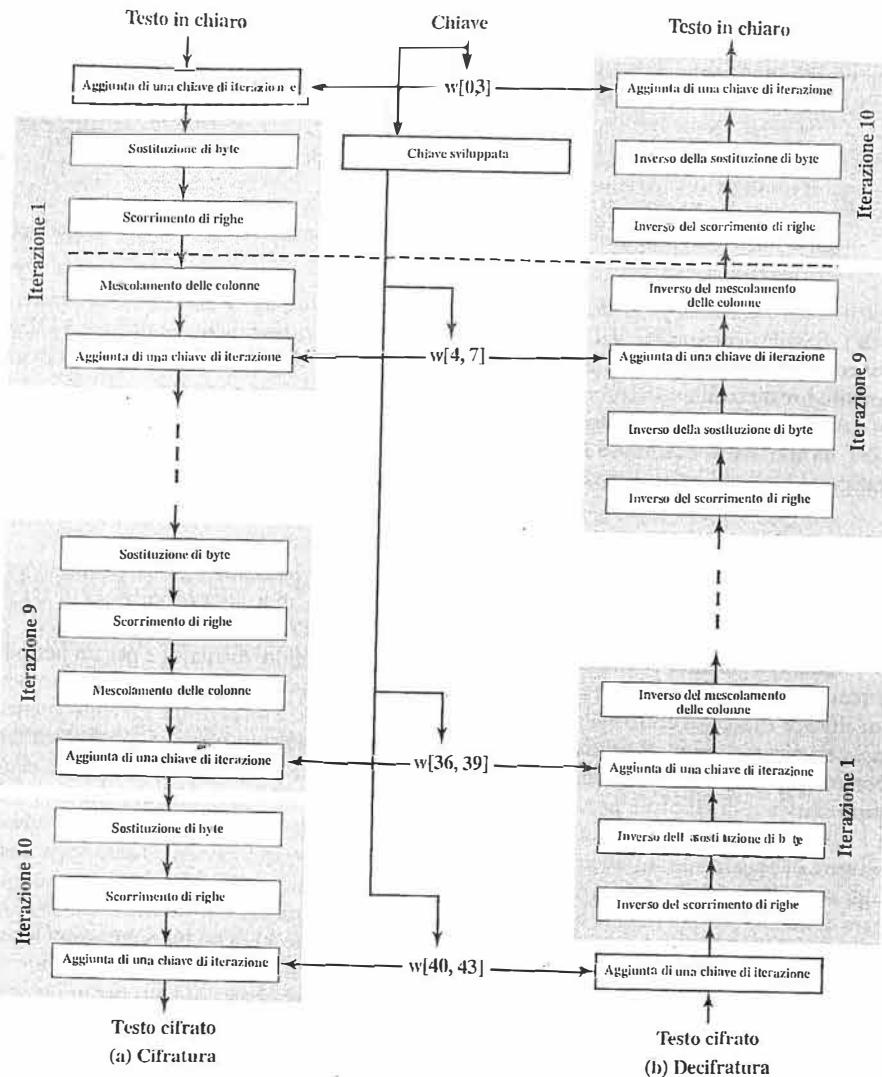


Figura 2.5 Cifratura e decifratura AES.

di valutazione e pubblicò lo standard definitivo (FIPS PUB 197) nel novembre del 2001. NIST selezionò Rijndael come proposta di algoritmo AES. I ricercatori che svilupparono e sottomisero Rijndael per AES sono due crittografi del Belgio: Joan Daemen e Vincent Rijmen.

Descrizione generale dell'algoritmo

AES usa un blocco della lunghezza di 128 bit e una lunghezza della chiave che può essere di 128, 192 o 256 bit. Nella descrizione di questo paragrafo si assume che la lunghezza della chiave sia di 128 bit, che è probabilmente la più comunemente implementata.

La Figura 2.5 mostra la struttura complessiva di AES: l'ingresso degli algoritmi di cifratura e decifratura è un singolo blocco di 128 bit, che in FIPS PUB 197 è rappresentato come una matrice quadrata di byte. Questo blocco viene copiato nell'array **Stato**, che viene modificato in ciascuno stadio di cifratura e decifratura; dopo lo stadio finale, **Stato** viene copiato in una matrice di uscita. In modo simile, la chiave di 128 bit viene rappresentata come una matrice quadrata di byte; questa chiave viene poi sviluppata in un array di parole che forma una tabella della chiave: ciascuna parola è di 4 byte e la tabella totale della chiave è di 44 parole per la chiave a 128 bit. L'ordinamento dei byte nella matrice è per colonne, così, ad esempio, i primi 4 byte di un testo in chiaro di 128 bit in ingresso al codice di cifratura occupa la prima colonna della matrice **in**, i secondi 4 byte occupano la seconda colonna e così via. In modo analogo, i primi 4 byte della chiave sviluppata, che formano una parola, occupano la prima colonna della matrice **w**.

Le seguenti osservazioni permettono di comprendere AES.

1. Una caratteristica importante è che non è una struttura di Feistel. Si ricordi che in una struttura di Feistel classica, la metà del blocco di dati viene usata per modificare l'altra metà del blocco di dati e poi le due metà vengono scambiate. AES non usa una struttura di Feistel, ma elabora tutto il blocco di dati in parallelo durante ciascun iterazione, usando sostituzioni e trasposizioni.
2. La chiave che viene fornita come ingresso viene sviluppata in un array di 44 parole a 32 bit, $w[i]$. Quattro parole distinte (128 bit) servono come chiave di iterazione per ciascuna iterazione.
3. Vengono usati quattro stadi differenti, uno di trasposizione e tre di sostituzione.
 - **Sostituzione di byte:** usa una tabella, detta S-box⁴, per eseguire la sostituzione byte a byte del blocco.
 - **Scorrimento di righe:** una semplice trasposizione che viene eseguita riga per riga.
 - **Mescolamento delle colonne:** una sostituzione che cambia ciascun byte in una colonna come una funzione di tutti i byte nella colonna.
 - **Aggiunta di una chiave di iterazione:** una semplice operazione logica sui bit di tipo XOR del blocco corrente con una porzione della chiave sviluppata.
4. La struttura è abbastanza semplice. Sia per la cifratura che la decifratura, il cifrario inizia con uno stadio **aggiunta di una chiave di iterazione** (*add round key*), seguito da nove iterazioni, ciascuna delle quali è costituita da quattro stadi, seguiti da una decima iterazione di tre stadi. Nella Figura 2.6 è rappresentata la struttura di un'intera iterazione di cifratura.

⁴ Il termine S-box, o scatola di sostituzione, è comunemente usato nella descrizione dei cifrari simmetrici per riferirsi a una tabella che viene impiegata per un tipo di ricerca in tabella del meccanismo di sostituzione.

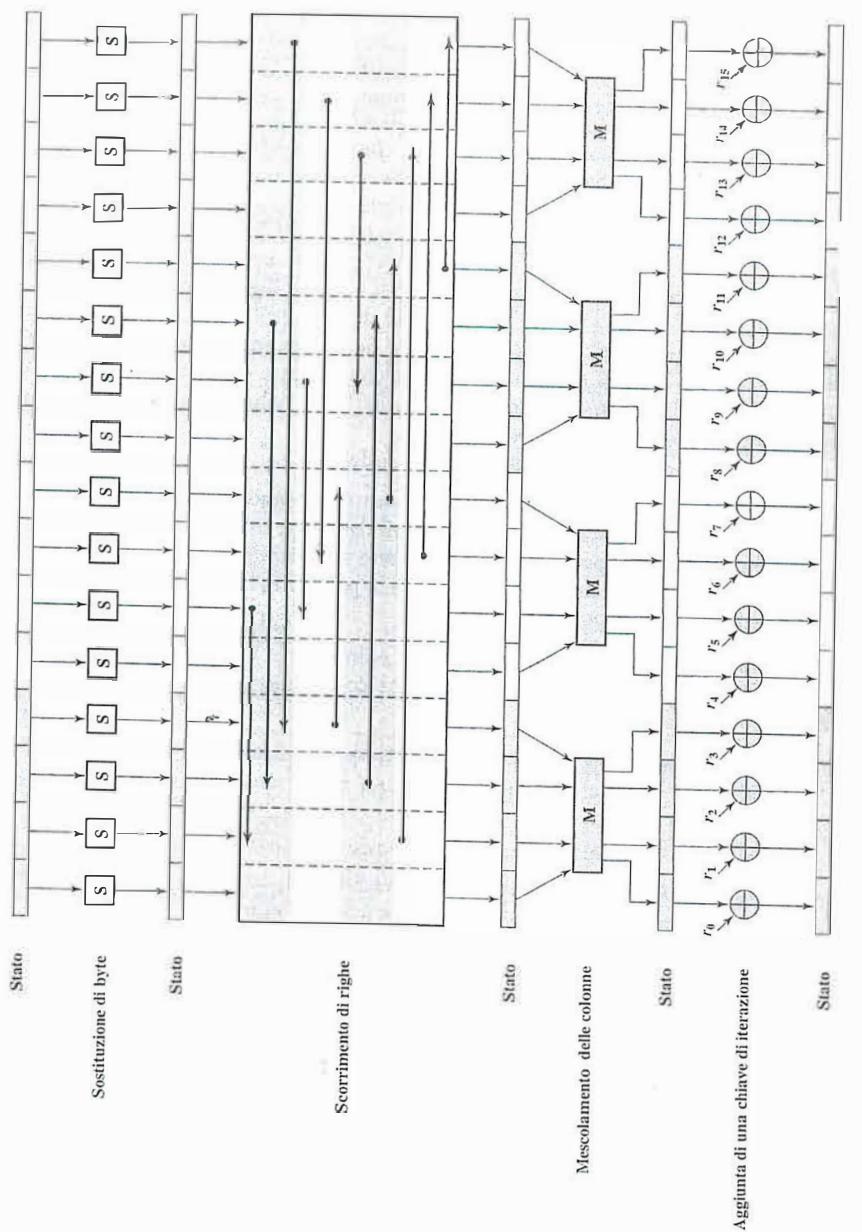


Figura 2.6 Iterazione di cifratura AES.

5. Solo lo stadio “aggiunta di una chiave di iterazione” fa uso della chiave. Per questa ragione, il cifrario inizia e finisce con uno stadio di aggiunta di una chiave di iterazione. Tutti gli altri stadi, applicati all’inizio o alla fine sono reversibili senza la conoscenza della chiave e, quindi, non aggiungerebbero sicurezza.
6. Lo stadio “aggiunta di una chiave di iterazione” da solo non sarebbe straordinario. Gli altri tre stadi insieme rimescolano i bit, ma da soli non fornirebbero sicurezza in quanto non usano la chiave. Si può vedere il cifrario come un’alternanza di operazioni di cifratura XOR (aggiunta di una chiave di iterazione) di un blocco, seguita da un rimescolamento del blocco (gli altri tre stadi), seguiti da una cifratura XOR e così via. Questo schema è efficiente e notevolmente sicuro.
7. Ciascuno stadio è facilmente reversibile. Per la “sostituzione dei byte”, lo “spostamento delle righe” e il “mescolamento delle colonne” viene usata una funzione inversa nell’algoritmo di decifratura. Per lo stadio di “aggiunta di una chiave di iterazione” l’ inverso si ottiene mettendo in XOR la stessa chiave di iterazione al blocco, usando il risultato che $A \oplus A \oplus B = B$.
8. Come la maggior parte dei cifrari a blocchi, l’algoritmo di decifratura utilizza la chiave sviluppata in ordine inverso. Tuttavia, l’algoritmo di decifratura non è identico all’algoritmo di cifratura: questa è una conseguenza della struttura particolare di AES.
9. Una volta che si è stabilito che tutti i quattro stadi sono reversibili, è facile verificare che la decifratura recupera il testo in chiaro. La Figura 2.5 mostra la cifratura e la decifratura che vanno in direzioni verticali opposte. A ciascun punto orizzontale (ad esempio, la linea tratteggiata nella figura), lo Stato è lo stesso per la cifratura e la decifratura.
10. L’iterazione finale di cifratura e decifratura consiste di tre soli stadi. Di nuovo, questa è una conseguenza della struttura particolare di AES ed è necessario per rendere il cifrario reversibile.

2.3 Cifrari a flusso e RC4

Un **cifrario a blocchi** elabora un blocco di elementi in ingresso per volta, producendo un blocco di uscita per ciascun blocco in ingresso. Un **cifrario a flusso** elabora continuamente gli elementi in ingresso, producendo in uscita un elemento per volta, man mano che procede. Sebbene i cifrari a blocchi siano molto più comuni, esistono alcune applicazioni nelle quali un cifrario a flusso è più adatto. Esempi verranno forniti successivamente in questo libro. In questo paragrafo si considererà quello che forse è il più popolare cifrario simmetrico a flusso, RC4. Si inizierà con una panoramica della struttura di un cifrario a flusso e poi si esaminerà RC4.

Struttura di un cifrario a flusso

Un tipico cifrario a flusso cifra il testo in chiaro un byte per volta, sebbene possa essere progettato per agire anche su unità più grandi di un byte per volta. La Figura 2.7 è un diagramma rappresentativo della struttura di un cifrario a flusso. In questa struttura una chiave è l’in-

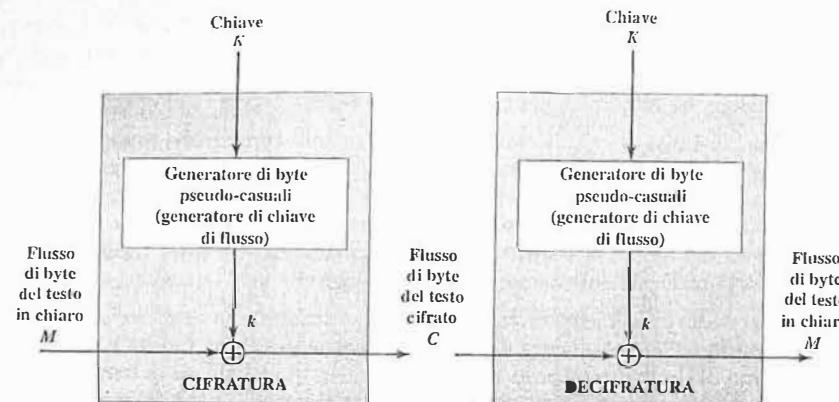


Figura 2.7 Diagramma di un cifrario a flusso.

gresso di un generatore pseudo-casuale di bit che produce un flusso di numeri a 8 bit che sono apparentemente casuali. Un flusso pseudo-casuale è un flusso che è “impredicibile” senza la conoscenza della chiave in ingresso, e che ha apparentemente caratteri casuali. L’uscita del generatore, chiamato *keystream* (**chiave di flusso**), è combinato un byte alla volta con il flusso del testo in chiaro usando un’operazione logica sui bit di OR esclusivo (XOR). Ad esempio, se il successivo bit originato dal generatore è 01101100 è il successivo byte del testo in chiaro è 11001100, il byte del testo cifrato risultante è:

$$\begin{array}{r} 11001100 \quad \text{testo in chiaro} \\ \oplus 01101100 \quad \text{chiave di flusso} \\ \hline 10100000 \quad \text{testo cifrato} \end{array}$$

La decifratura richiede l’uso della stessa sequenza pseudo-casuale:

$$\begin{array}{r} 10100000 \quad \text{testo in chiaro} \\ \oplus 01101100 \quad \text{chiave di flusso} \\ \hline 11001100 \quad \text{testo cifrato} \end{array}$$

Il cifrario a flusso è simile al riempimento, eseguibile una sola volta, discusso nel Capitolo 2. La differenza è che quel riempimento utilizza un vero flusso di numeri casuali, mentre il cifrario a flusso usa un flusso di numeri pseudo-casuali.

In [KUMA97] sono elencate le seguenti considerazioni progettuali importanti per un cifrario a flusso.

1. La sequenza di cifratura dovrebbe avere un periodo grande. Un generatore di numeri pseudo-casuali usa una funzione che genera un flusso deterministico di bit, che alla fine si ripete. Più è lungo il periodo di ripetizione, più difficile sarà la crittoanalisi.
2. La chiave di flusso dovrebbe approssimare il più possibile le proprietà di un vero flusso di numeri casuali. Ad esempio, ci dovrebbe essere approssimativamente un numero uguale di 0 e di 1. Se la chiave di flusso viene trattata come un flusso di byte, allora

Cifrario	Lunghezza della chiave	Velocità variabile (Mbps)
DES	56	9
3DES	168	3
RC2	variabile	0,9
RC4	variabile	45

Tabella 2.3 Confronto delle velocità di cifrari simmetrici su un Pentium II.

tutti i 256 possibili valori di byte dovrebbero approssimativamente apparire con una frequenza elevata e in modo uniforme. Più è casuale l’aspetto della chiave di flusso, più è randomizzato il testo cifrato, rendendo la crittoanalisi più difficile.

3. Si noti nella Figura 2.7 che l’uscita di un generatore di numeri pseudo-casuali è condizionata dal valore della chiave in ingresso. Per proteggersi dagli attacchi a forza bruta, è necessario che la chiave sia sufficientemente lunga. Le stesse considerazioni che si usano per i cifrari a blocchi, sono valide qui. Con la tecnologia attuale, quindi, è preferibile una lunghezza della chiave almeno di 128 bit.

Con un generatore di numeri pseudo-casuali correttamente progettato, un cifrario a flusso può essere sicuro tanto quanto un cifrario a blocchi con una lunghezza di chiave comparabile. Il vantaggio principale di un cifrario a flusso è che quasi sempre è più veloce e usa molto meno codice di quanto facciano i cifrari a blocchi. L’esempio in questo paragrafo, RC4, può essere implementato in appena poche linee di codice. Nella Tabella 2.3, usando i dati di [RESC01], si confrontano i tempi di esecuzione di RC4 con 3 ben noti cifrari simmetrici a blocchi. Il vantaggio di un cifrario a blocchi è che si possono riutilizzare le chiavi. Tuttavia, se due testi in chiaro sono cifrati con la stessa chiave, usando un cifrario a flusso, la crittoanalisi è spesso molto semplice [DAWS96]. Se i due flussi di testo cifrato sono messi in XOR tra loro, il risultato è lo XOR dei due testi in chiaro originali. Se i due testi in chiaro sono stringhe testuali, numeri di carta di credito o altri flussi di byte con proprietà note, allora la crittoanalisi può avere successo.

Per applicazioni che richiedono la cifratura/decifratura di un flusso di dati, come su un canale di comunicazione o su un collegamento browser/Web, un cifrario a flusso potrebbe essere l’alternativa migliore. Per applicazioni che hanno a che fare con blocchi di dati – come trasferimento di file, posta elettronica e database – un cifrario a blocchi può essere più adatto. Tuttavia, entrambi i tipi di cifrari possono virtualmente essere utilizzati in qualsiasi applicazione.

L’algoritmo RC4

RC4 è un cifrario a flusso, progettato nel 1987 da Ron Rivest per la RSA Security. È un cifrario a flusso con dimensione della chiave variabile e con un funzionamento orientato ai byte. L’algoritmo si basa sull’uso di una permutazione casuale. Le analisi mostrano che il periodo del cifrario è probabile che sia più grande di 10^{100} [ROBS95a]. Sono necessarie da otto a sedici operazioni macchina per un byte di uscita e ci si aspetta che il cifrario giri molto velocemente in software. RC4 viene usato negli standard SSL/TLS (*secure sockets layer/transport layer security*) definiti per le comunicazioni tra browser web e server. Viene

anche utilizzato nel protocollo WEP (*wired equivalent privacy*) e nel più recente protocollo WPA (*WiFi protected access*), questi protocolli sono entrambi degli standard delle LAN wireless IEEE 802.11. RC4 fu mantenuto come segreto commerciale della RSA Security e, nel settembre 1997, fu anonimamente inviato tramite Internet sulla lista del remailer anonimo Cypherpunks.

L'algoritmo RC4 è straordinariamente semplice e abbastanza agevole da spiegare. Una chiave di lunghezza variabile da 1 a 256 byte (da 8 a 2048 bit) viene utilizzata per inizializzare un vettore di stato a 256 byte, con elementi $S[0], S[1], \dots, S[255]$. In tutti i momenti, S contiene una permutazione di tutti i numeri a 8 bit da 0 a 255. Per la cifratura e decifrazione, viene generato un byte k (Figura 2.7) da S selezionando uno dei 255 elementi in maniera sistematica. Quando viene generato un valore di k , gli elementi in S vengono permessi ancora una volta.

Inizializzazione di S

Per cominciare, gli elementi di S sono impostati uguali ai valori da 0 a 255 in ordine crescente, cioè $S[0] = 0, S[1] = 1, \dots, S[255] = 255$. Viene anche creato un vettore temporaneo T . Se la lunghezza della chiave K è 256 byte, allora K viene trasferito in T ; altrimenti, per una chiave di lunghezza pari a $keylen$ byte, i primi $keylen$ elementi di T sono ottenuti come copia da K , e poi K è ripetuto tante volte quanto è necessario per riempire T . Queste operazioni preliminari possono essere riassunte come segue.

```
/* Inizializzazione */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

Successivamente si userà T per produrre la permutazione iniziale di S . Questo comporta il partire con $S[0]$, andare avanti fino a $S[255]$ e, per ciascun $S[i]$, scambiare $S[i]$ con un altro byte in S , a seconda dello schema dettato da $T[i]$:

```
/* Permutazione iniziale di S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Scambia (S[i], S[j]);
```

Poiché la sola operazione su S è lo scambio, il solo effetto è una permutazione: S contiene ancora tutti i numeri da 0 a 255.

Generazione del flusso

Una volta che il vettore S è inizializzato, la chiave di ingresso non viene più usata. La generazione del flusso viene realizzata ciclando su tutti gli elementi $S[i]$ e, per ciascun $S[i]$, scambiando $S[i]$ con un altro byte in S a seconda di uno schema dettato dalla configurazione corrente di S . Dopo che è stato raggiunto $S[255]$, il processo continua ripartendo da capo da $S[0]$.

```
/* Generazione del flusso */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Scambia (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

Per cifrare, si mette in XOR il valore k con il successivo byte del testo in chiaro, mentre per decifrare si mette in XOR il valore k con il successivo byte del testo cifrato. Nella Figura 2.8 è illustrata la logica di RC4.

Robustezza di RC4

Parecchi articoli sono stati pubblicati sull'analisi dei metodi per attaccare RC4 (ad esempio: [KNUD98], [MIST98], [FLUH00], [MANT01], [PUDO02], [PAUL03], [PAUL04]). Nessuno di questi approcci è praticabile contro RC4 con una lunghezza di chiave ragionevole, come 128 bit. Un problema più serio viene riportato in [FLUH01] dove gli autori dimostrano che il protocollo WEP, che si prefigge di fornire riservatezza sulle reti LAN wireless 802.11, è vulnerabile a un particolare approccio di attacco. Fondamentalmente il problema non è RC4, ma il modo con cui le chiavi sono generate per essere usate come ingresso a RC4. Questo problema particolare non sembra essere rilevante per altre applicazioni che usano RC4, e può essere risolto in WEP, cambiando il modo con cui le chiavi vengono generate. Questo problema mette in luce la difficoltà nel progettare un sistema sicuro che coinvolge sia le funzioni crittografiche sia i protocolli che le usano.

2.4 Modalità operative dei cifrari a blocchi

Un cifrario simmetrico a blocchi elabora un blocco di bit di dati alla volta. Nel caso di DES e 3DES, la lunghezza del blocco è 64 bit. Se la dimensione del testo in chiaro è maggiore, è necessario suddividere il testo in chiaro in blocchi di 64 bit (completando l'ultimo blocco se necessario). Il modo più semplice di procedere è la modalità nota come *electronic codebook mode* (ECB), in cui il testo in chiaro è elaborato in blocchi di 64 bit alla volta, e ogni blocco di testo in chiaro è cifrato utilizzando la stessa chiave. Il termine *codebook* deriva dal fatto che, per una data chiave, esiste un unico testo cifrato per ogni blocco di 64 bit di testo in chiaro. Quindi, ci si può immaginare un enorme dizionario (*codebook*) contenente una voce per ogni possibile sequenza di 64 bit di testo in chiaro. Questa voce contiene il corrispondente testo cifrato.

Con la modalità ECB, se lo stesso blocco di 64 bit di testo in chiaro compare più di una volta nel messaggio, produrrà sempre lo stesso testo cifrato. A causa di questa caratteristica, la modalità ECB può non essere sicura per messaggi lunghi. Se il messaggio è molto strutturato, può essere possibile per un crittoanalista sfruttare queste regolarità. Ad esempio, sapendo che il messaggio inizia sempre con un certo campo pre-definito, il crittoanalista avrà a disposizione un certo numero di testi in chiaro e corrispondenti testi cifrati su cui basarsi. Se il messaggio presenta elementi ripetuti, con un periodo di ripetizione multiplo di 64 bit,

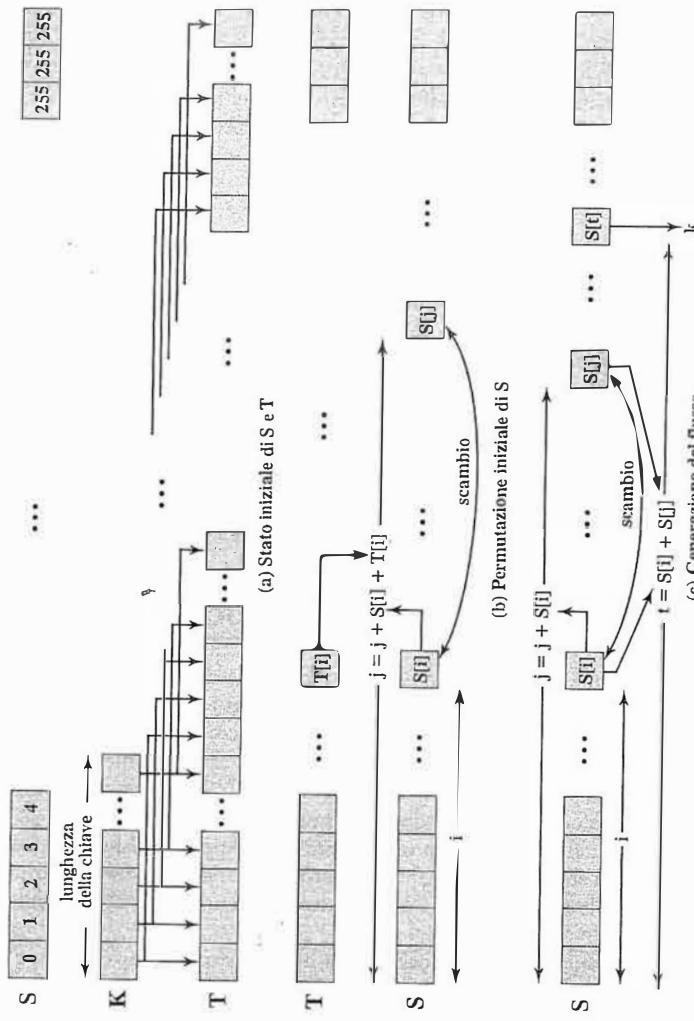


Figura 2.8 RC4.

allora questi elementi possono essere identificati dall'analista. Questo può rendere più agevole l'analisi o può fornire un'occasione per sostituire o riorganizzare i blocchi.

Per superare le carenze di sicurezza di ECB, è necessaria una tecnica in cui lo stesso blocco di testo in chiaro, se ripetuto, produca blocchi di testo cifrato differenti. In questo paragrafo, si esaminano due comuni alternative, definite in FIPS PUB 81.

Modalità cipher block chaining

Nella modalità *cipher block chaining* (CBC), illustrata nella Figura 2.9, l'ingresso dell'algoritmo di cifratura è costituito dall'OR esclusivo del blocco di testo in chiaro attualmente in ingresso e del blocco cifrato corrispondente al precedente blocco in ingresso; si utilizza la stessa chiave per ciascun blocco. In effetti, l'elaborazione delle sequenze di blocchi di testo in chiaro è stata concatenata. L'ingresso della funzione di cifratura per ogni blocco di testo in chiaro non ha alcuna relazione fissa con il blocco di testo in chiaro. Questo implica che non sono svelate sequenze ripetute di 64 bit.

Per quanto riguarda la decifratura, ciascun blocco di testo cifrato è elaborato mediante l'algoritmo di decifratura. Viene quindi effettuato l'OR esclusivo tra ciò che è restituito dall'algoritmo e il precedente blocco di testo cifrato, al fine di ottenere il blocco di testo in chiaro. Per comprendere meglio il funzionamento dell'algoritmo, sia:

$$C_j = E(K, [C_{j-1} \oplus P_j])$$

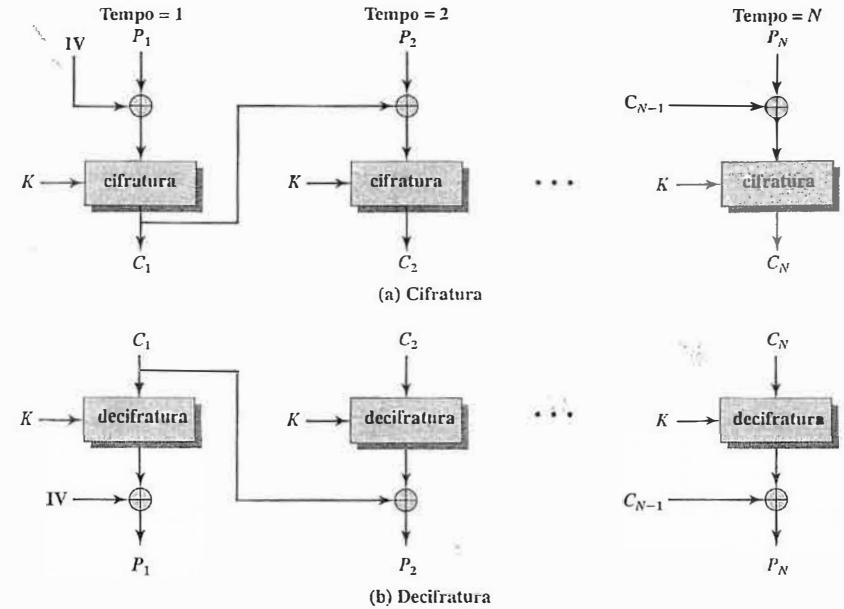


Figura 2.9 Modalità cipher block chaining (CBC).

dove $E[K, X]$ è la cifratura del testo in chiaro X mediante la chiave K , e \oplus rappresenta l'operazione di OR esclusivo. Allora:

$$\begin{aligned} D(K, C_j) &= D(K, E(K, [C_{j-1} \oplus P_j])) \\ D(K, C_j) &= C_{j-1} \oplus P_j \\ C_{j-1} \oplus D(K, C_j) &= C_{j-1} \oplus C_{j-1} \oplus P_j = P_j \end{aligned}$$

che dimostra quanto illustrato nella Figura 2.9b.

Per ottenere il primo blocco di testo cifrato si combina, mediante la funzione di OR esclusivo, un vettore di inizializzazione (IV, *initialization vector*) con il primo blocco di testo in chiaro. Durante la decifratura, per ottenere il primo blocco di testo in chiaro, IV è combinato mediante la funzione di OR esclusivo con il blocco risultante dall'algoritmo di decifratura.

IV deve essere conosciuto sia dal mittente che dal ricevente. Per massimizzare la sicurezza, IV deve essere ben protetto quanto la chiave. Tale protezione può essere realizzata spedendo IV mediante la modalità di cifratura ECB. Una delle ragioni per proteggere IV è la seguente: se un avversario è in grado di far utilizzare dal ricevente un valore diverso per IV, allora è in grado di invertire bit selezionati del primo blocco di testo in chiaro. Per convincersi di questo, si consideri la seguente formula:

$$\begin{aligned} C_1 &= E(K, [IV \oplus P_1]) \\ P_1 &= IV \oplus D(K, C_1) \end{aligned}$$

Si assume che $X[j]$ rappresenti il j -esimo bit dei 64 bit di X . Sia:

$$P_1[i] = IV[i] \oplus D(K, C_1)[i]$$

Usando le proprietà dell'operazione di OR esclusivo, si può affermare che:

$$P_1[i]' = IV[i]' \oplus D(K, C_1)[i]$$

dove l'apice denota il complemento di bit. Questo implica che se un avversario può modificare bit di IV in modo prevedibile, i corrispondenti bit del valore ricevuto per P_1 possono essere a loro volta modificati.

CBC è ampiamente utilizzato in applicazioni di sicurezza, come sarà illustrato nella seconda parte di questo libro.

Modalità cipher feedback

È possibile convertire qualsiasi cifrario a blocchi in un cifrario a flusso utilizzando il metodo *cipher feedback* (CFB). Con un cifrario a flusso non è necessario operare su messaggi la cui dimensione sia necessariamente un multiplo della dimensione del blocco, evitando quindi operazioni di completamento dei blocchi, ed è possibile anche operare in tempo reale. Ciò significa che, quando viene trasmesso un flusso di caratteri, ciascun carattere può essere cifrato e trasmesso immediatamente tramite un cifrario a flusso orientato ai caratteri.

Una caratteristica auspicabile per un cifrario a flusso è che la lunghezza del testo cifrato sia uguale a quella del testo in chiaro. Di conseguenza, se si trasmettono caratteri rappresentati mediante 8 bit, la cifratura di ciascun carattere deve richiedere esattamente 8 bit. L'utilizzo di un numero maggiore di bit, comporterebbe uno spreco della potenza di trasmissione.

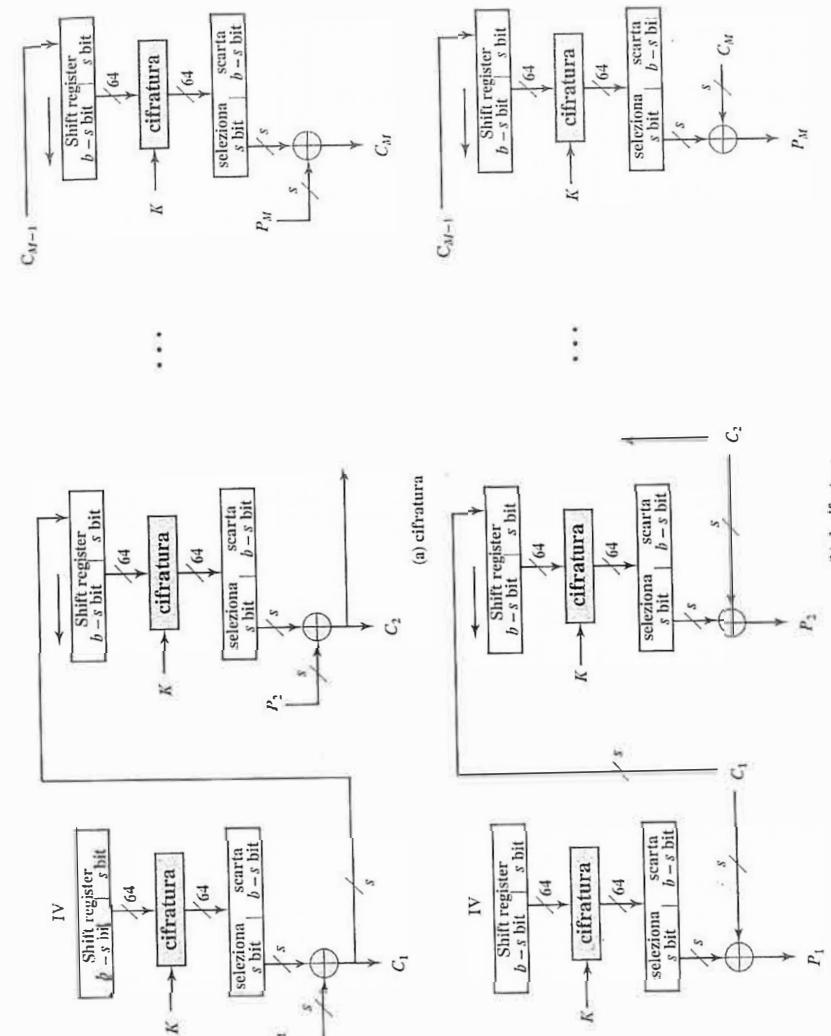


Figura 2.10 Modalità cipher feedback (CFB) a s-bit.

La modalità CFB è illustrata nella Figura 2.10, a pagina precedente, dove si ipotizza un'unità di trasmissione pari a s bit; un valore comune per s è 8. In analogia con la modalità CBC, le unità di testo in chiaro sono concatenate, in modo che il testo cifrato corrispondente a una qualsiasi unità di testo in chiaro sia funzione di tutto il testo in chiaro che precede tale unità.

Si consideri in primo luogo la cifratura. L'ingresso della funzione di cifratura è un registro di shift di dimensione 64 bit, il cui contenuto viene inizialmente posto uguale al valore di un certo vettore di inizializzazione (IV). Gli s bit più a sinistra (cioè quelli più significativi) del risultato della funzione di cifratura sono combinati mediante la funzione di OR esclusivo con la prima unità del testo in chiaro P_1 , per generare la prima unità di testo cifrato C_1 , che è successivamente trasmessa. Inoltre, il contenuto del registro di shift viene traslato a sinistra di s bit e C_1 è posto negli s bit più a destra (cioè quelli meno significativi) del registro stesso. Questo procedimento viene iterato fino a quando tutto il testo in chiaro è stato cifrato.

Per la decifratura si utilizza lo stesso schema, ad eccezione del fatto che l'unità di testo cifrato ricevuta viene combinata mediante la funzione di OR esclusivo con il risultato della funzione di cifratura, al fine di ottenere la corrispondente unità di testo in chiaro. Si noti come in questo contesto venga utilizzata la funzione di cifratura, e non quella di decifratura. Questo modo di operare ha una semplice spiegazione. Si denoti con $S_s(X)$ gli s bit più significativi di X . Allora si ha:

$$C_1 = P_1 \oplus S_s[E(K, IV)]$$

Da cui:

$$P_1 = C_1 \oplus S_s[E(K, IV)]$$

Lo stesso ragionamento vale per i passi successivi del procedimento.

2.5 Posizionamento dei dispositivi di cifratura

La cifratura è il metodo più efficace e comunemente utilizzato per contrastare possibili minacce alla sicurezza delle reti. Per utilizzare la cifratura, è necessario decidere cosa deve essere cifrato e dove posizionare il dispositivo di cifratura. A tal proposito esistono essenzialmente due alternative: la cifratura a livello di collegamento (*link encryption*) e la cifratura a livello mittente-ricevente (*end-to-end encryption*); la Figura 2.11 illustra l'utilizzo di entrambe queste modalità in una rete a commutazione di pacchetto.

Se si utilizza la cifratura a livello di collegamento, viene posizionato un dispositivo di cifratura ad entrambe le estremità di ciascuna linea di comunicazione giudicata vulnerabile. Questo implica che il traffico su tutte le linee di comunicazione è reso sicuro. Sebbene questo approccio abbia lo svantaggio di richiedere, su reti di dimensioni elevate, l'utilizzo di un consistente numero di dispositivi di cifratura, fornisce comunque un elevato livello di sicurezza. Uno degli svantaggi di questo approccio è che i messaggi devono essere decifrati tutte le volte che attraversano un commutatore di pacchetto; questa operazione è necessaria in quanto il commutatore deve essere in grado di leggere l'indirizzo (numero di circuito virtuale) contenuto nell'intestazione del pacchetto per poterlo instradare correttamente. La de-

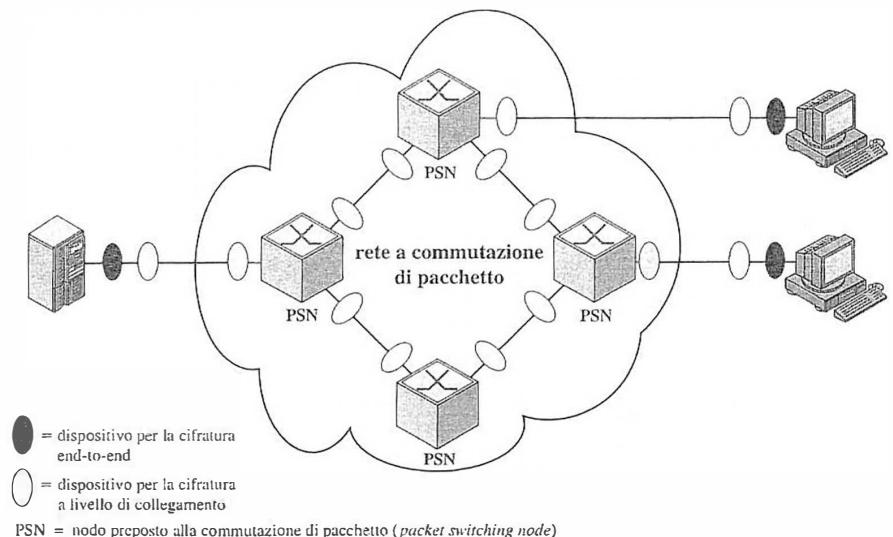


Figura 2.11 Cifratura su una rete a commutazione di pacchetto.

codifica rende quindi il messaggio vulnerabile tutte le volte che attraversa un commutatore. Se la rete è una rete pubblica a commutazione di pacchetto, l'utente non può esercitare alcun controllo sulla sicurezza dei nodi.

Con la cifratura a livello mittente-ricevente, invece, il processo di cifratura è a carico dei due sistemi terminali. L'host o il terminale sorgente cifra i dati che sono quindi trasmessi sulla rete all'host o terminale ricevente, senza subire ulteriori trasformazioni. Il ricevente condivide una chiave con il mittente e quindi è in grado di decifrare i dati. Questo approccio sembrerebbe rendere sicura la trasmissione contro attacchi sia alle linee di comunicazione sia ai commutatori. Anche questo approccio ha comunque ancora un punto debole.

Si consideri infatti il seguente scenario. Un host si connette a una rete X.25 a commutazione di pacchetto, instaura un circuito virtuale con un altro host, ed è quindi pronto a trasferire a quest'ultimo dati mediante la cifratura a livello mittente-ricevente. I dati sono trasmessi sulla rete sotto forma di pacchetti, che consistono in un'intestazione e in dati dell'utente. Quale porzione di ciascun pacchetto sarà cifrata dall'host? Si supponga che l'host cifri l'intero pacchetto, inclusa l'intestazione. Questo tipo di approccio non funziona, dal momento che solo l'host ricevente può effettuare la decifratura. Il nodo preposto alla commutazione del pacchetto riceverà un pacchetto cifrato; non sarà quindi in grado di leggere l'intestazione né, di conseguenza, potrà instradarlo correttamente. Pertanto, l'host può cifrare solo la porzione del pacchetto che contiene i dati utente, mentre deve lasciare l'intestazione in chiaro, in modo che possa essere letta dai dispositivi di rete.

Questo implica che, con la cifratura a livello mittente-ricevente, i dati utente sono sicuri. Al contrario, non lo è il flusso del traffico in quanto le intestazioni dei messaggi sono trasmesse in chiaro. Per ottenere un miglior grado di sicurezza sono necessarie sia la cifra-

tura a livello di collegamento sia la cifratura a livello mittente-ricevente, come illustrato nella Figura 2.11.

In sintesi, utilizzando entrambe le forme di cifratura, l'host cifra la porzione del pacchetto contenente i dati utente con una chiave di cifratura end-to-end. L'intero pacchetto è quindi cifrato utilizzando una chiave di cifratura a livello di collegamento. Man mano che il pacchetto viaggia sulla rete, ogni commutatore decifra il pacchetto utilizzando la chiave di cifratura a livello di collegamento per leggere l'intestazione del messaggio, e quindi cifra il pacchetto di nuovo prima di effettuarne la spedizione al dispositivo successivo. In questo modo l'intero pacchetto è sicuro in ogni momento ad eccezione del tempo in cui il pacchetto risiede nella memoria di un commutatore di pacchetti, in quanto durante tale periodo l'intestazione del pacchetto è in chiaro.

2.6 Distribuzione delle chiavi

Affinché la cifratura simmetrica sia efficace, è necessario che le due parti coinvolte nella comunicazione condividano la stessa chiave, e che questa sia protetta al fine di evitare accessi di terze parti. È inoltre consigliabile cambiare frequentemente le chiavi in modo da limitare i danni nel caso in cui qualcuno riesca ad impossessarsene. La robustezza di un sistema crittografico si basa quindi sulle tecniche adottate per la distribuzione delle chiavi, cioè sugli strumenti impiegati per distribuire le chiavi a due parti che vogliono scambiarsi dati, senza consentire ad altri di entrare in possesso della chiave. La distribuzione delle chiavi può essere effettuata in molti modi. Siano A e B le due parti che vogliono comunicare:

1. la chiave può essere scelta da A e materialmente consegnata a B;
2. la chiave può essere scelta da una terza parte che la consegna materialmente sia ad A che a B;
3. se A e B hanno recentemente condiviso una chiave, uno dei due può trasmettere una nuova chiave all'altro, cifrata con la vecchia chiave;
4. se A e B dispongono di una connessione cifrata con una terza parte C, C può rilasciare ad A e B una chiave utilizzando la cifratura a livello di collegamento.

Le opzioni 1 e 2 richiedono la consegna a mano della chiave. Nel caso si utilizzi la cifratura a livello di collegamento, questa è una richiesta ragionevole, in quanto ogni dispositivo di cifratura a livello di collegamento scambia dati solo con il suo partner all'altra estremità della linea. Per la cifratura a livello mittente-ricevente la consegna manuale delle chiavi è invece di difficile realizzazione. In un sistema distribuito, ciascun host o terminale può, nel corso del tempo, avere la necessità di effettuare scambi di dati con molti altri host e terminali. Ogni dispositivo necessita quindi di un certo numero di chiavi, fornite dinamicamente. Il problema è particolarmente complesso in un sistema distribuito su reti geografiche.

L'opzione 3 è praticabile sia per la cifratura a livello di collegamento sia per la cifratura a livello mittente-ricevente, ma se un avversario riuscisse ad accedere a una chiave, tutte le chiavi successive gli sarebbero rivelate. Anche se le chiavi usate per la cifratura a livello di

collegamento sono cambiate frequentemente, queste modifiche devono essere effettuate manualmente. L'opzione 4 è preferibile per fornire chiavi in sistemi basati su cifratura a livello mittente-ricevente.

La Figura 2.12 illustra un'implementazione della cifratura a livello mittente-ricevente basata sull'opzione 4. Nella figura non si considera la cifratura a livello di collegamento. La modalità a livello di collegamento può essere aggiunta o meno, a seconda delle specifiche necessità. Nello schema della Figura 2.12 vengono utilizzate due differenti tipologie di chiavi.

- **Chiave di sessione.** Quando due parti (host, terminali, etc.) desiderano comunicare, stabiliscono una connessione logica (per esempio, un circuito virtuale). Per la durata di questa connessione logica, tutti i dati utente sono cifrati con una chiave di sessione utilizzata una sola volta (*one-time key*). Alla fine della sessione o della connessione la chiave di sessione viene distrutta.
- **Chiave permanente.** Una chiave permanente è una chiave usata tra entità allo scopo di distribuire chiavi di sessione.

Lo schema della Figura 2.12 è costituito dalle seguenti entità.

- **Centro per la distribuzione di chiavi.** Il centro per la distribuzione di chiavi (KDC, *key distribution center*) decide quali sistemi possono comunicare tra loro. Quando a due sistemi viene concesso il permesso di stabilire una connessione, il centro per la distribuzione di chiavi fornisce loro una chiave di sessione per quella connessione.
- **Modulo del servizio di sicurezza.** (SSM, *security service module*) Questo modulo che può essere composto da una funzionalità a un livello protocollore, esegue la cifratura end-to-end (da punto finale a punto finale) e ottiene le chiavi di sessione per conto degli utenti.

I passi necessari per stabilire una connessione sono illustrati nella Figura 2.12. Quando un host desidera stabilire una connessione con un altro host, trasmette un pacchetto di richiesta connessione (passo 1). SSM salva il pacchetto e richiede a KDC il permesso di stabilire la connessione (passo 2). La comunicazione tra SSM e KDC è cifrata utilizzando una chiave master condivisa solo tra SSM e KDC. Se KDC concede il permesso di effettuare la connessione, genera la chiave di sessione e la recapita ai due SSM appropriati, usando un'unica chiave permanente per ciascun SSM (passo 3). SSM che ha effettuato la richiesta può a questo punto rilasciare il pacchetto di richiesta di connessione. Si stabilisce così una connessione tra i due sistemi terminali (passo 4). Tutti i dati utente scambiati tra i due sistemi terminali sono cifrati dai corrispondenti SSM, impiegando la precedente chiave di sessione.

La distribuzione automatizzata delle chiavi possiede quelle caratteristiche di dinamicità e flessibilità necessarie per consentire a numerosi utenti di accedere a diversi host e per consentire a questi ultimi il reciproco scambio di dati.

Un ulteriore approccio alla distribuzione di chiavi, basato sulla cifratura a chiave pubblica, è presentato nel Capitolo 3.

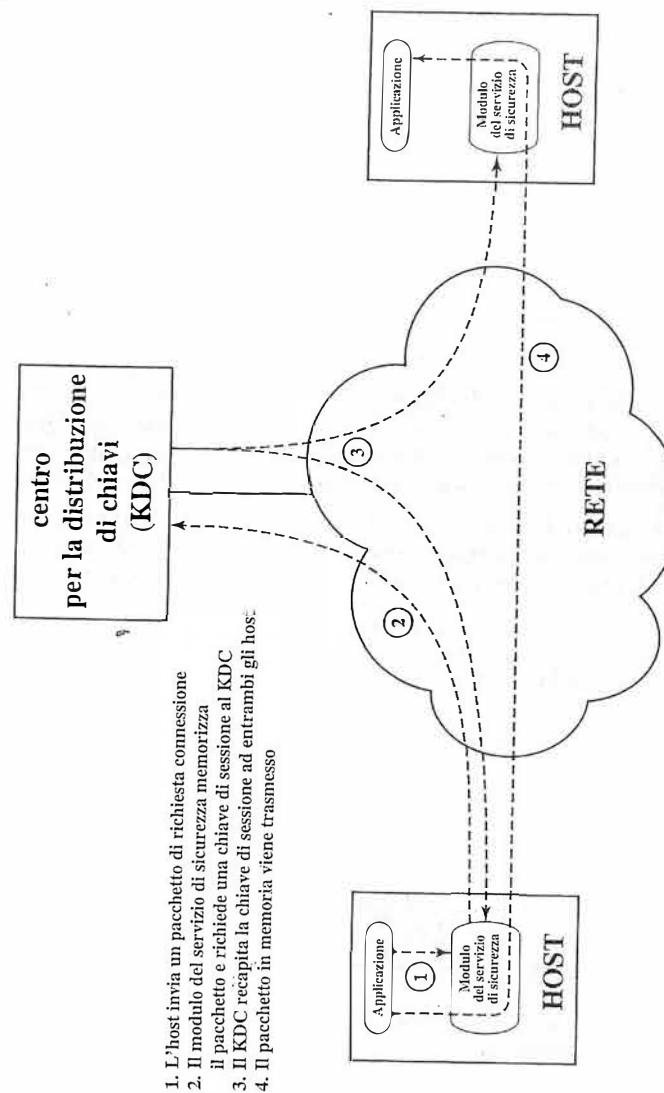


Figura 2.12 Distribuzione automatizzata di chiavi per protocolli orientati alla trasmissione.

2.7 Letture consigliate e siti web

Gli argomenti di questo capitolo sono trattati molto dettagliatamente in [STAL06a]. Per quanto riguarda gli algoritmi crittografici, [SCHN96] costituisce un fondamentale testo di consultazione che contiene una descrizione di quasi tutti gli algoritmi e i protocolli di cifratura pubblicati fino al momento della stesura del libro. Un'altra interessante e dettagliata panoramica è [MENE97]. Una trattazione più approfondita, rigorosamente matematica, è [STIN06].

Siti web consigliati

- **Home page di AES:** la pagina del NIST su AES. Contiene lo standard oltre a parecchi altri documenti rilevanti.
- **AES Lunge:** contiene un'esauriente bibliografia di documenti e articoli su AES, con accesso alle copie elettroniche.
- **Block cipher mode of operation:** pagina del NIST con informazioni complete sulle modalità operative approvate dal NIST.

2.8 Domande di revisione ed esercizi

Domande di revisione

- 2.1 Quali sono gli ingredienti essenziali di un cifrario simmetrico?
- 2.2 Quali sono le due funzioni di base usate in un algoritmo di cifratura?
- 2.3 Quante chiavi sono necessarie per due persone che comunicano tramite un cifrario simmetrico?
- 2.4 Qual è la differenza tra un cifrario a blocchi e un cifrario a flusso?
- 2.5 Quali sono i due approcci generali per violare un cifrario?
- 2.6 Perché alcuni modi di funzionamento di un cifrario a blocchi usano solo la cifratura, mentre altri usano cifratura e decifratura?
- 2.7 Che cos'è la cifratura tripla?
- 2.8 Perché la porzione centrale di 3DES è una decifratura piuttosto che una cifratura?
- 2.9 Qual è la differenza tra la cifratura di collegamento e quella end-to-end?
- 2.10 Elenca i modi con cui le chiavi segrete possono essere distribuite tra le due parti comunicanti.
- 2.11 Qual è la differenza tra la chiave di sessione e la chiave master?
- 2.12 Che cos'è il centro di distribuzione delle chiavi?

Esercizi

- 2.1 Dimostrate che la decifratura mediante il cifrario di Feistel è l'inverso della corrispondente cifratura.
- 2.2 Quale valore della chiave RC4 lascerà S non modificato durante l'inizializzazione? Cioè, dopo la permutazione iniziale di S, gli elementi di S saranno uguali ai valori da 0 fino a 255 in ordine crescente.

- 2.3 RC4 ha uno stato interno segreto che è una permutazione di tutti i possibili valori del vettore S e dei due indici i e j .
- Usando uno schema semplice per memorizzare il suo stato interno, quanti bit vengono usati?
 - Immaginate di considerarlo dal punto di vista di quante informazioni sono rappresentate dallo stato. In quel caso, è necessario determinare quanti stati differenti ci sono e poi prendere il logaritmo in base 2, per trovare quanti bit di informazione questo rappresenta. Usando questo approccio quanti bit sarebbero necessari per rappresentare lo stato?
- 2.4 Eseguendo DES in modalità ECB, se esiste un errore in un blocco del testo cifrato trasmesso, questo influenza solo il corrispondente blocco di testo in chiaro. Nella modalità CBC, invece, l'errore viene propagato. Ad esempio, un errore in C_1 (Figura 2.9) altera ovviamente anche P_1 e P_2 .
- Ci sono dei blocchi oltre P_2 che sono colpiti?
 - Supponete che esista un errore a livello di bit nella versione sorgente di P_1 . Quanti sono i blocchi di testo cifrato in cui questo errore si propaga? Quali sono le conseguenze sul ricevente?
- 2.5 CBC-Pad è un metodo di funzionamento di un cifrario a blocchi, utilizzato nel cifrario a blocchi RC5, ma può essere usato in qualunque cifrario a blocchi. CBC-Pad gestisce i testi in chiaro di qualunque lunghezza e il testo cifrato è più lungo del testo in chiaro al massimo della dimensione di un singolo blocco. Il riempimento viene impiegato per garantire che il testo in chiaro in ingresso sia un multiplo della lunghezza del blocco. Si è ipotizzato che il testo in chiaro originale sia un numero intero di byte. Questo testo in chiaro viene riempito alla fine da 1 a bb byte, dove bb è uguale alla dimensione del blocco in byte. I byte di riempimento sono tutti uguali e impostati al byte che rappresenta il numero di byte del riempimento. Ad esempio, se vi sono 8 byte di riempimento, ciascun byte avrà la configurazione dei bit 00001000. Perché non permettere 0 byte di riempimento? Cioè, se il testo in chiaro originale è un intero multiplo della dimensione del blocco, perché non evitare di riempire?
- 2.6 Il riempimento potrebbe non essere sempre adatto. Ad esempio, si potrebbe voler memorizzare i dati cifrati nello stesso buffer di memoria che originariamente conteneva il testo in chiaro. In questo caso il testo cifrato deve essere della stessa lunghezza del testo in chiaro originale. Un modo per raggiungere questo scopo è il metodo *ciphertext stealing* (CTS). Nella Figura 2.13a è rappresentata un'implementazione di questo metodo.
- Spiegate come funziona.
 - Descrivete come decifrare C_{n-1} e C_n .
- 2.7 La Figura 2.13b mostra un'alternativa a CTS per produrre un testo cifrato delle stesse dimensioni del testo in chiaro, quando quest'ultimo non è un intero multiplo delle dimensioni del blocco.
- Spiegate l'algoritmo.
 - Spiegate perché CTS è preferibile a questo approccio illustrato nella Figura 2.13b.
- 2.8 Supponete che un errore a livello di bit avvenga durante la trasmissione di un carattere cifrato quando si adotta la modalità CFB a 8 bit. Fino a che punto si propaga l'errore?
- 2.9 Gli schemi di distribuzione delle chiavi che utilizzano un centro di controllo dell'accesso e/o un centro per la distribuzione di chiavi sono basati su un approccio centralizzato che è quindi vulnerabile ad attacchi. Discutete le implicazioni rispetto alla sicurezza che tale centralizzazione comporta.

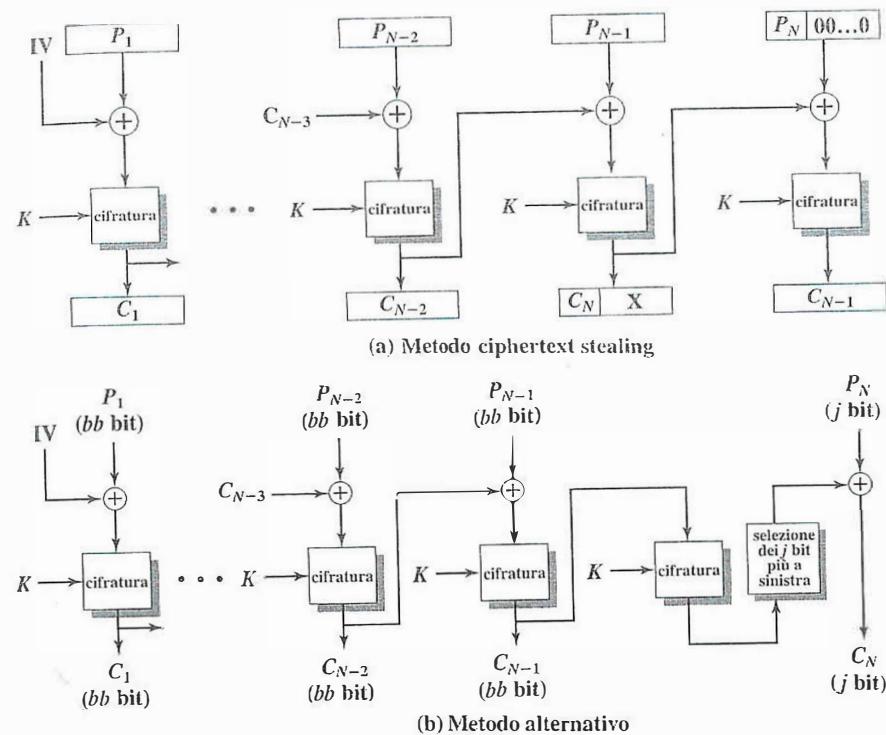


Figura 2.13 Metodi di cifratura a blocchi del testo in chiaro che non è multiplo delle dimensioni del blocco.

- 2.10 Supponete che venga suggerito il seguente metodo per assicurarsi del fatto che due parti siano entrambe in possesso della stessa chiave segreta. Una parte genera una stringa di bit casuali di lunghezza pari alla lunghezza della chiave, poi la compone mediante la funzione di OR esclusivo con la chiave, e spedisce il risultato sul canale di comunicazione. L'altra parte combina, tramite la funzione di OR esclusivo, il blocco ricevuto con la chiave (che deve essere la stessa posseduta dall'altra parte) e spedisce il risultato all'altra parte che verifica quanto ricevuto; se coincide con la stringa casuale generata all'inizio, allora è sicura che l'altra parte possiede la stessa chiave segreta, anche se nessuna delle due parti ha mai trasmesso tale chiave. Stabilite se il metodo proposto presenta qualche punto debole.

Capitolo 3

Crittografia a chiave pubblica e autenticazione dei messaggi

L'autenticazione dei messaggi è un'importante funzione di sicurezza delle reti che va ad aggiungersi a quella di riservatezza. Questo capitolo prende in considerazione tre aspetti connessi all'autenticazione dei messaggi. Prima di tutto, si considera l'impiego di codici di autenticazione di messaggio e funzioni hash per fornirne l'autenticazione; successivamente, si esamineranno i principi di cifratura a chiave pubblica e due specifici algoritmi a chiave pubblica utili anche per lo scambio di chiavi nella cifratura convenzionale. In seguito, si descriverà l'adozione della cifratura a chiave pubblica per produrre firme digitali, realizzando così un livello di autenticazione dei messaggi ancora più elevato. Infine, verrà riconsiderato il problema della gestione delle chiavi di cifratura.

3.1 Approcci all'autenticazione dei messaggi

La cifratura protegge da attacchi passivi (spionaggio). La protezione da attacchi attivi (falsificazione di dati e transazioni) richiede differenti requisiti ed è conosciuta come **autenticazione dei messaggi**.

Un messaggio, un file, un documento o qualunque altro insieme di dati si dice autentico quando non è stato alterato e proviene effettivamente dalla sorgente dichiarata. L'autenticazione dei messaggi è una procedura che consente alle parti in comunicazione di verificare che i messaggi ricevuti siano autentici. I due aspetti principali riguardano la verifica del fatto che i contenuti del messaggio non siano stati alterati e che la sorgente sia autentica. Inoltre, è possibile verificare la puntualità del messaggio (ovvero che il messaggio non è stato artificialmente ritardato e replicato) e la sequenza relativa rispetto ad altri messaggi scambiati fra due parti.

Autenticazione basata su cifratura convenzionale

È possibile effettuare l'autenticazione utilizzando semplicemente la cifratura convenzionale. Ipotizzando che solo il mittente e il ricevente condividano una chiave (come dovrebbe essere), soltanto il mittente originale sarebbe in grado di cifrare un messaggio in modo intellegibile per l'altra parte. Inoltre, se il messaggio contiene un codice di rilevazione di errore e un numero di sequenza, il ricevente potrebbe verificare che non siano state apportate mo-

difiche e che il numero di sequenza sia corretto. Se il messaggio contenesse anche un timestamp, il ricevente potrebbe verificare che questo non sia stato ritardato oltre il tempo normalmente richiesto per il transito lungo la rete.

Autenticazione dei messaggi senza cifratura del messaggio

In questo paragrafo prenderemo in esame alcuni approcci all'autenticazione dei messaggi, che non sfruttano tecniche di cifratura. In tutti questi approcci si genera un'etichetta di autenticazione, che è apposta a ciascun messaggio per la trasmissione. Il messaggio non viene cifrato e può essere letto dal ricevente indipendentemente dalla funzione di autenticazione da questi utilizzata.

Dato che gli approcci presentati in questo paragrafo non cifrano il messaggio, non si può garantirne la riservatezza. Dal momento che la cifratura convenzionale fornisce autenticazione, ed essendo ampiamente impiegata in prodotti facilmente disponibili, ci si potrebbe domandare perché non utilizzare semplicemente tale approccio, in grado di fornire sia riservatezza sia autenticazione. [DAVI89] identifica tre situazioni in cui è preferibile l'autenticazione del messaggio senza riservatezza.

1. In molte applicazioni lo stesso messaggio è inviato in modalità broadcast a un dato numero di destinatari (ad esempio, la segnalazione a utenti dell'indisponibilità della rete, oppure il segnale di allarme in una centrale di controllo). È meno costoso e più affidabile che sola destinazione sia responsabile del controllo dell'autenticità. Quindi, il messaggio deve essere inviato in modalità broadcast senza essere cifrato e con associata un'etichetta di autenticazione. Il sistema responsabile effettua l'autenticazione. Se si verifica una violazione alla sicurezza, gli altri sistemi destinatari sono allertati mediante una segnalazione generale di allarme.
2. Un altro scenario possibile è lo scambio di messaggi fra due parti in cui una è sovraccarica e non può dedicare tempo alla decifratura di tutti i messaggi in arrivo. L'autenticazione viene eseguita selettivamente, scegliendo in modo casuale i messaggi da controllare.
3. L'autenticazione di un programma in chiaro (non cifrato) è un servizio interessante. Il programma può essere eseguito senza doverlo ogni volta decifrare, evitando così spreco di risorse di elaborazione. Tuttavia, disponendo di un'etichetta di autenticazione associata al programma, è possibile controllare il programma ogniqualvolta sia necessario avere la certezza della sua integrità.

In conclusione, si può notare che sia l'autenticazione che la cifratura hanno un proprio ruolo al fine di soddisfare i requisiti di sicurezza.

Codice di autenticazione di messaggio

Una possibile tecnica di autenticazione dei messaggi fa uso di una chiave segreta per generare un piccolo blocco di dati, conosciuto come codice di autenticazione di messaggio (MAC, *message authentication code*), che viene accodato al messaggio stesso. Questa tecnica presuppone che le due parti comunicanti, A e B, condividano una chiave segreta K_{AB} . Quando A deve inviare un messaggio a B, provvede a calcolare il codice di autenticazione di messaggio

come funzione del contenuto del messaggio e della chiave: $MAC_M = F(K_{AB}, M)$. Il messaggio e il relativo codice sono quindi trasmessi al destinatario stabilito. Questi esegue la stessa elaborazione sul messaggio ricevuto, utilizzando la stessa chiave segreta per generare un nuovo codice di autenticazione di messaggio. Il codice ricevuto insieme al messaggio viene quindi confrontato con il codice calcolato dal ricevente (Figura 3.1). Supponendo che solo il mittente e il ricevente conoscano la chiave segreta e che il codice ricevuto coincida con quello calcolato, allora è possibile trarre le seguenti conclusioni.

1. Il ricevente ha garanzia che il messaggio non è stato alterato. Se un avversario modificasse il contenuto del messaggio ma non il codice ad esso associato, il codice calcolato dal ricevente differirebbe da quello ricevuto insieme al messaggio. Avendo ipotizzato che l'avversario non sia a conoscenza della chiave segreta, non è possibile per lui modificare il codice di autenticazione di messaggio in modo da rispecchiare le modifiche apportate al contenuto del messaggio.
2. Il ricevente ha garanzia che il messaggio proviene dal mittente dichiarato. Dato che nessun altro è a conoscenza della chiave segreta, nessun altro potrebbe creare un messaggio avente proprio quel codice.
3. Se il messaggio contiene un numero di sequenza (come accade, ad esempio, con i protocolli X.25, HDLC e TCP), allora il ricevente ha garanzia della correttezza della sequenza, in quanto un avversario non può modificarla.

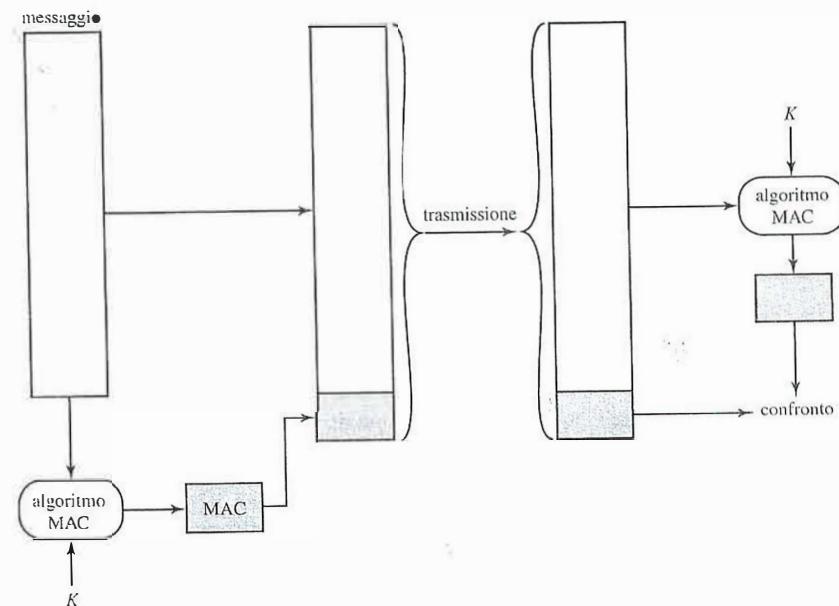


Figura 3.1 Autenticazione di messaggi mediante codice di autenticazione di messaggio (MAC).

Per generare il codice di autenticazione del messaggio si possono utilizzare svariati algoritmi. La specifica di NIST, FIPS PUB 113, raccomanda l'uso DES. Questo algoritmo viene impiegato per generare una versione cifrata del messaggio utilizzando, come codice, un certo numero di bit alla fine del testo cifrato. Un tipico codice è composta da 16 o 32 bit.

Questo processo presenta somiglianze con il processo di cifratura. La principale differenza è che il processo di autenticazione non deve necessariamente essere reversibile, come invece deve essere la decifratura. Inoltre, le proprietà matematiche della funzione di autenticazione fanno sì che sia meno vulnerabile agli attacchi rispetto alla funzione di cifratura.

Funzione hash unidirezionale

Un'alternativa al codice di autenticazione del messaggio è la funzione hash unidirezionale (*one-way*). Come nel caso del codice di autenticazione di messaggio, una funzione hash accetta in ingresso un messaggio di lunghezza variabile M e produce in uscita un digest di messaggio $H(M)$ di lunghezza fissa. A differenza del codice di autenticazione di messaggio, una funzione hash non necessita di una chiave segreta in ingresso. Per effettuare l'autenticazione dei messaggi, il digest di messaggio viene inviato insieme con il messaggio stesso.

La Figura 3.2 illustra tre modalità per l'autenticazione dei messaggi. Il digest di messaggio può essere cifrato utilizzando tecniche di cifratura convenzionale (a); l'autenticità è garantita ipotizzando che solo il mittente e il ricevente conoscano la chiave segreta di cifratura. Si può anche cifrare il contenuto del messaggio con tecniche di cifratura a chiave pubblica (b). L'argomento sarà approfondito nel Paragrafo 3.5. L'approccio a chiave pubblica ha due vantaggi: in primo luogo, fornisce sia firma digitale sia autenticazione del messaggio e, secondariamente, non richiede la distribuzione delle chiavi fra le parti che devono comunicare. Rispetto agli approcci che cifrano l'intero contenuto del messaggio, questi hanno il vantaggio di richiedere meno elaborazione. Tuttavia, si è comunque manifestato un particolare interesse nello sviluppo di tecniche che evitino del tutto la cifratura. Le principali motivazioni in tal senso sono descritte in [TSUD92].

- La cifratura software è piuttosto lenta. Anche se la quantità di dati da cifrare per messaggio è piccola, può esserci un flusso costante di messaggi in ingresso e in uscita da un sistema.
- La cifratura hardware ha costi non trascurabili. Sono disponibili implementazioni dell'algoritmo DES su chip a basso costo, ma le spese richieste aumentano se tutti i nodi della rete devono avere questa funzionalità.
- La cifratura hardware è ottimizzata in funzione di grandi quantitativi di dati. Per piccoli blocchi di dati, un'elevata percentuale del tempo è dedicata a procedure di inizializzazione/invocazione.
- Gli algoritmi di cifratura possono essere protetti da brevetto.

La Figura 3.2c mostra una tecnica per l'autenticazione dei messaggi che impiega una funzione hash senza cifratura. In questa tecnica si ipotizza che due parti comunicanti, A e B, condividano un valore segreto S_{AB} . Quando A deve inviare un messaggio a B, calcola la funzione hash sul risultato della concatenazione del valore segreto e del contenuto del messag-

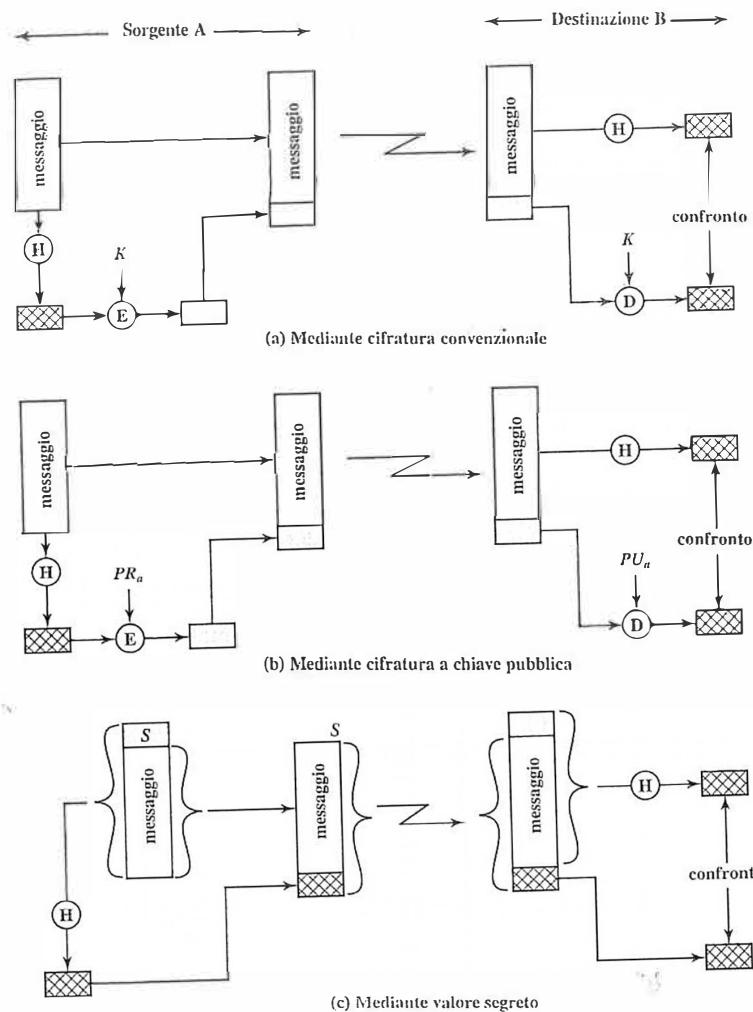


Figura 3.2 Autenticazione di messaggi mediante funzione hash unidirezionale.

gio: $MD_M = H(S_{AB} \parallel M)$ ¹. Successivamente il risultato $[M \parallel MD_M]$ viene inviato a B. Dato che B possiede S_{AB} , può ricalcolare $H(S_{AB} \parallel M)$ e verificare MD_M . Dal momento che il valore segreto non viene trasmesso, un avversario non può modificare il messaggio, qualora lo

¹ \parallel denota l'operazione di concatenazione.

intercetti. Fino a quando il valore segreto rimane tale, un avversario non può generare un messaggio falso.

Una variazione della terza tecnica, denominata HMAC, viene utilizzata per la sicurezza IP (Capitolo 6), ed è stata specificata anche per SNMPv3 (Capitolo 8).

3.2 Funzioni hash sicure e HMAC

La funzione hash unidirezionale, o funzione hash sicura, è rilevante non solo per l'autenticazione dei messaggi ma anche per la firma digitale. Questo paragrafo inizia con una descrizione dei requisiti di una funzione hash sicura. In seguito descriveremo una delle più importanti funzioni hash: SHA. Infine, esamineremo HMAC.

Requisiti di una funzione hash sicura

L'obiettivo di una funzione hash è produrre un'*impronta digitale* di un file, di un messaggio o, in generale, di un blocco di dati. Per essere utile nell'autenticazione di messaggi, una funzione hash H deve possedere le seguenti proprietà.

1. H può essere applicata a un blocco di dati di qualsiasi dimensione.
2. H produce in uscita un risultato di lunghezza fissa.
3. Dato un qualunque ingresso x , $H(x)$ è relativamente semplice da calcolare rendendo possibili nella pratica implementazioni a livello hardware e software.
4. Per qualunque valore b , è computazionalmente impossibile trovare x tale che $H(x) = b$. Qualche volta, in letteratura si fa riferimento a questa come alla **proprietà di unidirezionalità**.
5. Per un qualunque blocco x , è computazionalmente impossibile trovare $y \neq x$ con $H(x) = H(y)$. A ciò, qualche volta, si fa riferimento come alla **resistenza alle collisioni debole**.
6. È computazionalmente impossibile trovare una coppia (x, y) tale che $H(x) = H(y)$. A ciò, qualche volta, si fa riferimento come alla **resistenza alle collisioni forte**².

Le prime tre proprietà esprimono requisiti per l'utilizzo pratico di una funzione hash per l'autenticazione dei messaggi. La quarta è la cosiddetta proprietà di unidirezionalità per la quale è semplice generare un codice dato un messaggio, ma è potenzialmente impossibile derivare il messaggio dato il codice. Questa proprietà è importante quando la tecnica di autenticazione fa uso di un valore segreto (Figura 3.2c). Tale valore non viene inviato ma, se la funzione hash non fosse unidirezionale, un avversario potrebbe facilmente scoprirla; se

² Sfortunatamente, questi termini non sono usati in modo consistente. Termini alternativi, usati in letteratura, comprendono la funzione hash unidirezionale (proprietà 4 e 5), funzione hash resistente alle collisioni (proprietà 4, 5 e 6); funzione hash unidirezionale debole (proprietà 4 e 5); funzione hash unidirezionale forte (proprietà 4, 5 e 6). Il lettore deve fare attenzione nell'approcciarsi alla letteratura per determinare il significato del particolare termine usato.

l'avversario è in grado di osservare o intercettare una trasmissione, potrebbe ottenere il messaggio M e il codice hash $C = H(S_{AB} \| M)$. L'avversario potrebbe quindi invertire la funzione hash per ottenere $S_{AB} \| M = H^{-1}(C)$. Dato che l'avversario è ora in possesso sia di M sia di $S_{AB} \| M$, la ricostruzione di S_{AB} diventa banale.

La quinta proprietà garantisce che è impossibile trovare un messaggio alternativo avente lo stesso valore hash di un messaggio dato. Questa proprietà impedisce la contraffazione quando si utilizza un codice hash cifrato (Figura 3.2a e b). Se questa proprietà non fosse verificata, un avversario potrebbe essere in grado di agire secondo il seguente schema: primo, osservare o intercettare il messaggio e il suo codice hash cifrato; secondo, generare un codice hash non cifrato a partire dal messaggio; terzo, generare un messaggio alternativo avente lo stesso codice hash.

Una funzione hash che soddisfa le cinque proprietà sopra elencate è detta **funzione hash debole**. Se la funzione hash soddisfa anche la sesta proprietà, allora viene denominata **funzione hash forte**. La sesta proprietà offre garanzie di protezione nei confronti di una classe di attacchi sofisticati, noti come attacchi birthday. I dettagli di questo attacco sono al di fuori dell'ambito di questo libro. L'attacco riduce la forza di una funzione hash a m -bit da 2^m a $2^{m/2}$. Si veda [YUVA79] e [STAL06a] per informazioni più approfondite.

Oltre a fornire autenticazione, un digest di messaggio fornisce anche integrità dei dati in quanto svolge la stessa funzione della sequenza di controllo del frame: se qualche bit del messaggio venisse accidentalmente modificato durante la trasmissione, il digest di messaggio risulterebbe errato.

Funzioni hash semplici

Tutte le funzioni hash operano sulla base dei seguenti principi generali. I dati in ingresso (ad esempio, un messaggio o un file) sono considerati come una sequenza di blocchi di n bit. I dati in ingresso sono elaborati un blocco alla volta iterativamente per produrre una funzione hash di n bit.

Una delle più semplici funzioni hash è quella che esegue l'OR esclusivo (XOR) bit a bit di ciascun blocco. Questo può essere espresso come segue:

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

dove

C_i indica l' i -esimo bit del codice hash, $1 \leq i \leq n$

m indica il numero di blocchi di n bit ricevuti in ingresso dalla funzione

b_{ij} indica l' i -esimo bit all'interno del j -esimo blocco

\oplus indica l'operazione di OR esclusivo

Questa modalità di esecuzione è mostrata nella Figura 3.3; tale funzione hash produce un semplice bit di parità per ciascuna posizione dei bit, ed è conosciuta come **controllo di redundanza longitudinale**. È ragionevolmente efficace per effettuare controlli di integrità su dati senza un formato riconoscibile. Ciascun valore a n bit generato dalla funzione hash è equiprobabile. Quindi, la probabilità che un errore nei dati produca lo stesso valore nel corrispondente valore hash è 2^{-n} . Per dati in cui si riconosce un certo formato, questa funzione è meno efficace. Ad esempio, nei più comuni file di testo, il bit più alto di **ciascun byte** è

	bit 1	bit 2	• • •	bit n
Blocco 1	b_{11}	b_{21}		b_{n1}
Blocco 2	b_{12}	b_{22}		b_{n2}
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
Blocco m	b_{1m}	b_{2m}		b_{nm}
Codice hash	C_1	C_2		C_n

Figura 3.3 Semplice funzione hash che utilizza l'OR esclusivo bit a bit.

sempre zero. Utilizzando un valore hash di 128 bit, l'efficacia della funzione hash con questo tipo di dati diventa 2^{-112} anziché 2^{-128} .

Un miglioramento consiste nell'eseguire, dopo aver elaborato ciascun blocco, uno spostamento circolare di un bit, o rotazione, nel valore hash generato. La procedura può essere sintetizzata come segue.

1. Inizialmente, porre a zero gli n bit del valore hash.
2. Elaborare ciascun blocco di dati di n bit in successione come segue:
 - a. effettuare una rotazione sinistra di un bit del valore hash attuale;
 - b. eseguire l'OR esclusivo del blocco per generare il valore hash.

Queste operazioni sortiscono l'effetto di rendere casuali in maniera più completa i dati in ingresso superando le regolarità che possono verificarsi negli stessi.

Anche se questa seconda procedura rappresenta una buona misura per l'integrità dei dati, può diventare potenzialmente inutile per la loro sicurezza quando si utilizza un codice hash cifrato insieme con un messaggio in chiaro (come mostrato nelle Figure 3.2a e b). Dato un messaggio, diventa facile generare un nuovo messaggio avente lo stesso codice hash: basta preparare il messaggio alternativo desiderato e quindi accodarvi un blocco di n bit che forzi il nuovo messaggio e il blocco ad avere il codice hash desiderato.

Anche se il semplice OR esclusivo o l'OR esclusivo con rotazione (RXOR) sono insufficienti se si effettua la cifratura del solo codice hash, queste semplici funzioni potrebbero comunque essere ritenute utili nel caso in cui vengano cifrati sia il messaggio sia il codice hash. Anche in questo caso occorre procedere con cautela. Una tecnica originariamente proposta dal National Bureau of Standards utilizzava il semplice OR esclusivo applicato a blocchi di 64 bit del messaggio e, quindi, effettuava la cifratura dell'intero messaggio secondo la modalità *cipher block chaining* (CBC). Questo schema può essere definito come segue: dato un messaggio composto da una sequenza di blocchi di 64 bit X_1, X_2, \dots, X_N , produrre il codice hash C effettuando l'OR esclusivo blocco a blocco oppure su tutti i blocchi e accodare il codice hash al messaggio come blocco finale del messaggio stesso:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \dots \oplus X_N$$

Successivamente, cifrare sia l'intero messaggio sia il codice hash secondo la modalità CBC per generare il messaggio cifrato Y_1, Y_2, \dots, Y_{N+1} . [JUEN85] mette in evidenza diversi modi

con cui è possibile manipolare il testo cifrato del messaggio ottenuto in questo modo, senza che tali modifiche siano riflesse nel codice hash. Ad esempio, dalla definizione di CBC (Figura 2.9), si ha che:

$$X_1 = IV \oplus D(K, Y_1)$$

$$X_i = Y_{i-1} \oplus D(K, Y_i)$$

$$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$$

Ma X_{N+1} è il codice hash:

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \dots \oplus X_N \\ &= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \dots \oplus [Y_{N-1} \oplus D(K, Y_N)] \end{aligned}$$

Dato che nell'equazione precedente è possibile eseguire l'OR esclusivo dei vari termini in un qualsiasi ordine, ne consegue che il codice hash non cambia a fronte di permutazioni dei blocchi cifrati.

La funzione hash sicura SHA-1

Il *secure hash algorithm* (SHA) fu sviluppato dal National Institute of Standards and Technology (NIST) e pubblicato come *Federal information processing standard* (FIPS 180) nel 1993; una versione modificata è stata pubblicata nel 1995 come FIPS 180-1, alla quale si fa generalmente riferimento con la sigla SHA-1. SHA-1 è specificato anche nella RFC 3174 che, fondamentalmente, è una copia del materiale presente in FIPS 180-1, cui aggiunge un'implementazione in codice C.

SHA-1 produce un valore hash di 160 bit. Nel 2002, NIST pubblicò una nuova versione dello standard, FIPS 180-2, nella quale venivano definiti tre nuove versioni di SHA con lunghezze dei valori hash di 256, 384 e 512 bit, note rispettivamente come SHA-256, SHA-384 e SHA-512 (Tabella 3.1). Queste nuove versioni si basano sulla stessa struttura e usano lo stesso tipo di aritmetica modulare e le stesse operazioni in logica binaria di SHA-1. Nel 2005, NIST annunciò l'intenzione di eliminare gradualmente l'approvazione di SHA-1. Nel 2010, NIST annunciò l'intenzione di eliminare gradualmente l'approvazione di SHA-1 e spingere al ricorso di altre versioni di SHA entro il 2010. Poco dopo, un gruppo

	SHA-1	SHA-256	SHA-384	SHA-512
Dimensione del digest del messaggio	160	256	384	512
Dimensione del messaggio	$< 2^{64}$	$< 2^{61}$	$< 2^{128}$	$< 2^{128}$
Dimensione del blocco	512	512	1024	1024
Dimensione della parola	32	32	64	64
Numero di passi	80	64	80	80
Sicurezza	80	128	192	256

Note: 1. Tutte le dimensioni sono misurate in bit.
2. La sicurezza si riferisce al fatto che un attacco birthday sul digest del messaggio di dimensione n produce una collisione con workfactor, approssimativamente pari a $2^{n/2}$.

Tabella 3.1 Confronto tra i parametri di SHA.

di ricerca descrisse un attacco nel quale si possono trovare due messaggi separati che rilasciano lo stesso hash SHA-1 usando 2^{69} operazioni, molte meno delle 2^{80} che precedentemente si pensava fossero necessarie per trovare una collisione con un hash SHA-1 [WANG05]. Questo risultato avrebbe dovuto accelerare la transizione verso le altre versioni di SHA.

In questo paragrafo viene descritto SHA-1, le altre versioni sono abbastanza simili. L'algoritmo prende come ingresso un messaggio con una lunghezza massima minore di 2^{128} bit e produce in uscita un digest del messaggio di 512 bit. Il messaggio in ingresso viene elaborato in blocchi di 1024 bit. Nella Figura 3.4 è rappresentata l'elaborazione completa di un messaggio per produrre il digest. L'elaborazione consiste nei seguenti passi.

- Passo 1. Aggiunta di bit di completamento.** Il messaggio viene completato con bit aggiuntivi in modo tale da rendere la sua lunghezza congruente al valore 896 modulo 1024 (lunghezza = 896 mod 1024). Il completamento viene sempre effettuato, anche nei casi in cui il messaggio ha già la lunghezza adatta. Di conseguenza, il numero di bit di completamento varia da 1 a 1024. Il completamento consiste in un primo bit con valore 1 seguito da tanti bit con valore 0 quanti sono quelli necessari a raggiungere la lunghezza corretta.
- Passo 2. Aggiunta della lunghezza.** In questo passo, si aggiunge un blocco di 128 bit in coda al messaggio. Questo blocco è trattato come un intero senza segno a 128 bit (con il byte più significativo in prima posizione) e contiene la lunghezza del messaggio originale (cioè prima di effettuare il completamento).

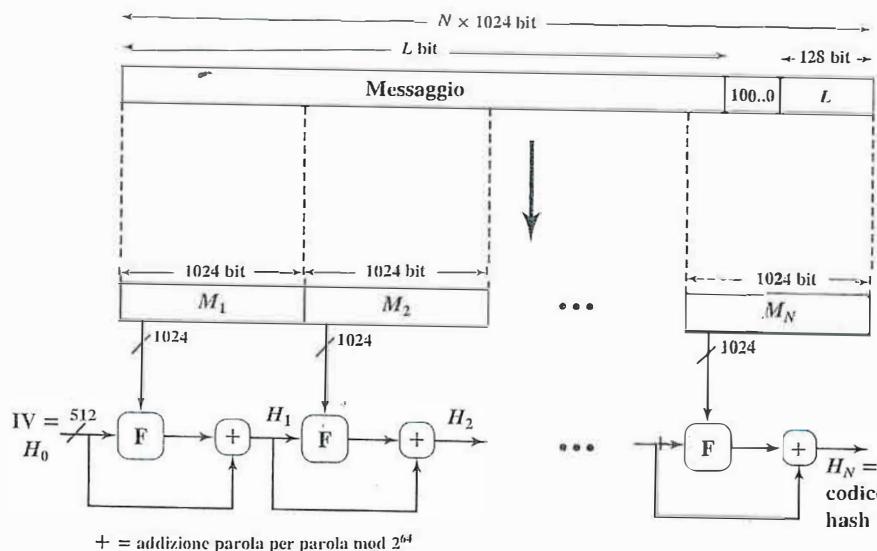


Figura 3.4 Generazione del digest di messaggio con SHA-512.

Il risultato di questi primi due passi produce un messaggio avente come lunghezza un multiplo di 1024 bit. Nella Figura 3.4, il messaggio espanso è rappresentato come una sequenza di blocchi di 1024 bit M_1, M_2, \dots, M_N in modo che la lunghezza totale del messaggio espanso è $N \times 1024$ bit.

- Passo 3. Inizializzazione del buffer hash.** Si utilizza un buffer di 512 bit per memorizzare sia il risultato intermedio sia quello finale della funzione hash. Il buffer può essere rappresentato come un insieme di otto registri a 64 bit (a, b, c, d, e, f, g, h). Questi registri sono inizializzati con i seguenti interi a 64 bit (valori esadecimali):

a = 6A09E667F3BCC908	e = 510E527FADE682D1
b = BB67AE8584CAA73B	f = 9B05688C2B3E6C1F
c = 3C6EF372FE94F82B	g = 1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1	h = 5BE0CD19137E2179

Questi valori sono memorizzati nel formato *big-endian*, in cui il byte più significativo di una parola è nella posizione del byte a indirizzo più basso (più a sinistra). Queste parole sono state ottenute prendendo i primi 64 bit della parte frazionaria delle radici quadrate dei primi 8 numeri primi.

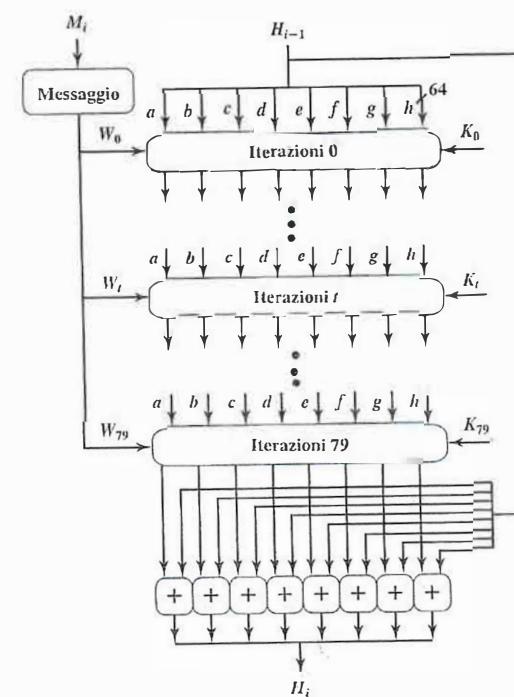


Figura 3.5 Elaborazione di un singolo blocco di 1024 bit con SHA-512.

- Passo 4. Elaborazione del messaggio in blocchi di 1024 bit** (parole di 128 byte). Il cuore dell'algoritmo è un modulo che consiste di 80 iterazioni. Questo modulo è indicato con F nella Figura 3.4, mentre la logica è illustrata nella Figura 3.5, a pagina precedente.

Ciascuna iterazione prevede in ingresso (*input*) il valore del buffer a 512 bit “abcdefg” e aggiorna il contenuto del buffer. All'ingresso della prima iterazione, il buffer ha un valore pari a quello del valore intermedio dell'hash, H_{i-1} . Ciascuna iterazione i fa uso di un valore W_i a 64 bit derivato dal blocco di 1024 bit che si sta elaborando in quel momento (M_i). Ciascuna iterazione, inoltre, utilizza una costante additiva K_i , dove $0 \leq i \leq 79$ indica una delle 80 iterazioni. Queste parole rappresentano i primi 64 bit della parte frazionaria delle radici cubiche dei primi 8 numeri primi. Le costanti forniscono un insieme di configurazioni casuali a 64 bit, che dovrebbero eliminare qualsiasi regolarità nei dati di ingresso.

Il risultato dell'ottantesima iterazione è aggiunto all'ingresso della prima iterazione (H_{i-1}) per produrre H_i . L'addizione viene eseguita indipendentemente per ciascuna delle otto parole nel buffer con ciascuna delle corrispondenti parole in H_{i-1} , usando l'addizione modulo 2^{64} .

- Passo 5. Risultato (*output*)**. Dopo che tutti gli N blocchi di 1024 bit sono stati elaborati, il risultato dell' N -esimo è il digest di 512 bit del messaggio.

L'algoritmo SHA-512 gode della proprietà che ciascun bit del codice hash è funzione di ogni bit dei dati in ingresso. La complessa ripetizione della funzione di base F produce buoni risultati; infatti, è improbabile che due messaggi scelti a caso, pur avendo apparentemente regolarità simili, abbiano lo stesso codice hash. A meno che esista qualche debolezza nascosta nell'algoritmo SHA-512, eventualità al momento non ancora pubblicata, per ottenere due messaggi con lo stesso digest è necessario un numero di operazioni dell'ordine di 2^{256} , mentre per ottenere un messaggio con un dato digest è necessario un numero di operazioni dell'ordine di 2^{512} .

Altre funzioni hash sicure

Come nel caso dei cifrari a blocchi simmetrici, i progettisti di funzioni hash sicure si sono dimostrati riluttanti a deviare da una struttura comprovata. L'algoritmo DES si basa sul cifrario di Feistel. Praticamente tutti i più importanti cifrari a blocchi successivi seguono o lo schema di Feistel o una sua generalizzazione che comporta un numero ancora maggiore cicli di funzioni di sostituzione e permutazione. Schemi di questo tipo possono essere adattati a resistere ai nuovi attacchi di crittoanalisi. Se per realizzare un cifrario a blocchi simmetrico si utilizzasse uno schema totalmente nuovo, non si potrebbe escludere che questo possa aprire nuove vie di attacco non ancora identificate. Analogamente, le più importanti funzioni hash oggi utilizzate seguono la struttura base mostrata nella Figura 3.4, nota come funzione hash iterata, originariamente proposta da Merkle [MERK79, MERK89]. Le motivazioni alla base della struttura iterativa derivano dalle considerazioni di Merkle [MERK89] e Damgard [DAMG89] relativamente al fatto che se la funzione per un singolo blocco, nota come **funzione di compressione**, è resistente alle collisioni, allora lo è la funzione hash iterata risultante. Pertanto, questa struttura può essere utilizzata al fine di generare una funzione hash sicura per operare su messaggi di qualsiasi lunghezza.

Il problema di sviluppare una funzione hash sicura si riduce a quello di progettare una funzione di compressione resistente alle collisioni che operi su dati in ingresso di una data lunghezza. Essendosi questo rivelato come un approccio sostanzialmente solido, i nuovi progetti si propongono semplicemente di raffinare questa struttura con aggiunte alla lunghezza del codice hash.

In questo paragrafo, in aggiunta alla funzione SHA vengono prese in considerazione altre due funzioni hash sicure, accettate in ambito commerciale.

Algoritmo MD5 per la generazione di digest di messaggio

L'algoritmo MD5 per la generazione di digest di messaggio (RFC 1321) è stato sviluppato da Ron Rivest. Fino a pochi anni fa, quando è sorto l'interesse verso gli attacchi a forza bruta e verso la crittoanalisi, MD5 era l'algoritmo hash sicuro più diffusamente utilizzato. Questo algoritmo riceve in ingresso un messaggio di lunghezza arbitraria e produce come risultato un digest di messaggio di 128 bit. I dati in ingresso sono elaborati a blocchi di 512 bit.

Con l'aumentare della velocità di elaborazione, la sicurezza offerta da un codice hash di 128 bit è diventata discutibile. Si può dimostrare che la complessità di determinare due messaggi aventi lo stesso digest è dell'ordine delle 2^{64} operazioni, mentre la complessità di trovare un messaggio avente un dato digest è dell'ordine delle 2^{128} operazioni. Questi sono numeri troppo piccoli per garantire sicurezza. Inoltre, è stata sviluppata una gamma di attacchi di crittoanalisi che mostrano come MD5 sia vulnerabile a questo tipo di aggressioni [BERS92, BOER93, DOBB96].

Whirlpool

Whirlpool [BARR03, STAL06b] fu sviluppato dal belga Vincent Rijmen – co-ideatore di Rijndael, adottato come Advanced Encryption Standard (AES) – e dal crittografo brasiliano Paulo Barreto. Whirlpool è una delle due funzioni hash approvate da NESSIE (*New European Schemes for Signatures, Integrity and Encryption*) [PREN02]³. Il progetto NESSIE è uno sforzo, sponsorizzato dalla Comunità Europea, di proporre un portafoglio di forti primitive crittografiche di vario tipo, inclusi i cifrari a blocchi, i cifrari simmetrici, le funzioni hash e i codici di autenticazione del messaggio.

Whirlpool è basato sull'uso di un cifrario a blocchi per la funzione di compressione, anzi Whirlpool usa un cifrario a blocchi che è stato specificatamente progettato per essere utilizzato nella funzione hash e che probabilmente non sarà mai impiegato come funzione di cifratura a se stante. Questo perché i progettisti volevano far uso di un cifrario a blocchi avente la sicurezza e l'efficienza di AES, ma con una lunghezza hash in grado di fornire una sicurezza potenziale pari a quella di SHA-512. Il risultato è il cifrario a blocchi W, che ha una struttura simile e usa le stesse funzioni elementari di AES, ma che utilizza una dimensione del blocco e della chiave pari a 512 bit.

HMAC

Recentemente, si è manifestato crescente interesse verso lo sviluppo di codici di autenticazione di messaggio (MAC) derivati da codice hash crittografico, cioè che utilizza la cifratura, come nel caso dell'algoritmo SHA-1. Questo è dovuto a diverse motivazioni.

³ L'altro schema approvato consiste nelle tre varianti di SHA: SHA-256, SHA-384 e SHA-512.

- Le funzioni hash che utilizzano la cifratura sono in genere eseguite in software più velocemente degli algoritmi di cifratura convenzionale come, ad esempio, DES.
- È ampiamente disponibile codice di libreria per funzioni hash che utilizzano la cifratura.

Non essendo stata progettata per scopi di MAC, una funzione hash come SHA-1 non può essere direttamente utilizzata per tali scopi, in quanto non sfrutta una chiave segreta. Sono state avanzate alcune proposte per incorporare una chiave segreta all'interno di un algoritmo hash esistente. Fra queste, l'approccio HMAC è quello che ha ricevuto maggiore sostegno [BELL96a, BELL96b]. HMAC, pubblicato come documento RFC 2104, è stato scelto come obbligatorio per la realizzazione di MAC per la sicurezza IP, ed è utilizzato in altri protocolli Internet quali, ad esempio, TLS (*transport layer security*, sicurezza del livello di trasporto) e SET (*secure electronic transaction*, transazioni elettroniche sicure).

Obiettivi di progetto di HMAC

Il documento RFC 2104 elenca i seguenti obiettivi progettuali di HMAC.

- Usare, senza modifiche, le funzioni hash disponibili (in particolare, funzioni hash con buone prestazioni software, il cui codice è disponibile gratuitamente e facilmente reperibile).
- Consentire un'agevole sostituzione della funzione hash contenuta, nel caso in cui siano trovate o richieste funzioni hash più veloci o più sicure.
- Preservare le prestazioni originarie della funzione hash senza incorrere in degradazioni significative delle prestazioni.
- Usare e gestire le chiavi in maniera semplice.
- Essere in possesso di un'analisi crittografica consolidata della "robustezza" del meccanismo di autenticazione, basata su ipotesi ragionevoli relative alla funzione hash contenuta.

I primi due obiettivi sono importanti per l'accettabilità di HMAC. HMAC tratta la funzione hash come una "scatola nera". Questo comporta due vantaggi. Primo, nella realizzazione di HMAC è possibile utilizzare un'implementazione esistente di una funzione hash come un modulo. In questo modo, gran parte del codice HMAC è preconfezionata e pronta all'uso senza modifiche. In secondo luogo, qualora si desiderasse sostituire la funzione hash in un'implementazione di HMAC, tutto ciò che è necessario fare è rimuovere la funzione hash esistente e passare al nuovo modulo. Questo si potrebbe fare, ad esempio, se si desiderasse passare a una funzione hash più veloce. Ma soprattutto, se la sicurezza della funzione hash fosse compromessa, si potrebbe mantenere la sicurezza di HMAC semplicemente sostituendo la funzione hash esistente con una funzione hash più sicura.

L'ultimo obiettivo di progetto della precedente lista presenta, in pratica, il vantaggio principale di HMAC rispetto ad altre proposte di schemi basati su funzioni hash. Se la funzione hash contenuta possiede ragionevoli punti di forza per quanto riguarda la crittografia, si può dimostrare che HMAC è sicuro. Questo aspetto verrà ripreso in seguito, dopo aver esaminato la struttura di HMAC.

Algoritmo HMAC

La Figura 3.6 illustra il funzionamento complessivo di HMAC. Si introducono le seguenti notazioni:

H = funzione hash contenuta (ad esempio, SHA-1)

M = messaggio in ingresso a HMAC (compreso il completamento specificato nella funzione hash)

Y_i = i -esimo blocco di M , $0 \leq i \leq (L - 1)$

L = numero di blocchi in M

b = numero di bit in un blocco

n = lunghezza del codice hash prodotto dalla funzione hash contenuta

K = chiave segreta; se la lunghezza della chiave è maggiore di b , la chiave viene data in ingresso alla funzione hash per produrre una chiave di n bit; la lunghezza consigliata è $\geq n$

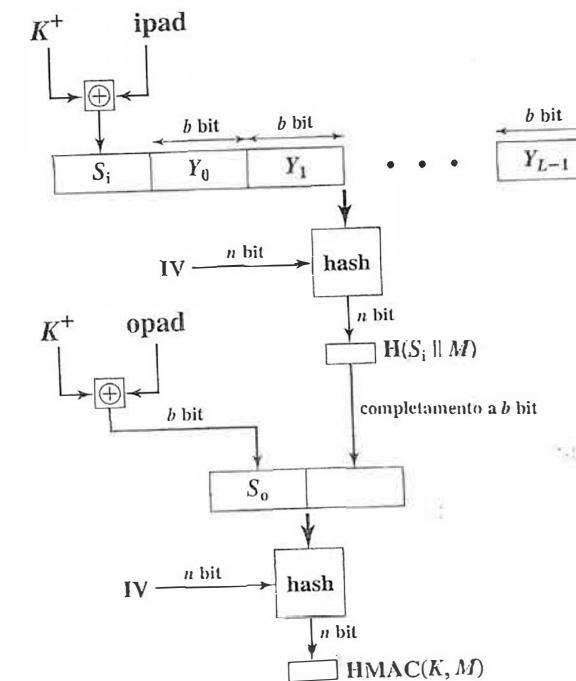


Figura 3.6 Struttura di HMAC.

$K^* = K$ completato con una serie di zero a sinistra in modo che il risultato abbia lunghezza b bit

ipad = 00110110 (36 in esadecimale) ripetuto $b/8$ volte

opad = 01011100 (5C in esadecimale) ripetuto $b/8$ volte

Allora HMAC si può esprimere come segue:

$$\text{HMAC}(K, M) = H[(K^* \oplus \text{opad}) \| H[(K^* \oplus \text{ipad}) \| M]]$$

Equivalentemente:

1. aggiungere una serie di zero alla fine della parte sinistra di K per creare una stringa di b bit K^* (ad esempio, se K ha lunghezza 160 bit e $b = 512$, allora si aggiungeranno a K 44 byte con valore zero);
2. eseguire l'OR esclusivo (OR esclusivo bit a bit) fra K^* e ipad per produrre il blocco di b bit S_i ;
3. accodare M a S_i ;
4. applicare H al flusso generato al Passo 3;
5. eseguire l'OR esclusivo fra K^* e opad per produrre il blocco di b bit S_o ;
6. accodare il risultato del Passo 4 a S_o ;
7. applicare H al flusso generato al Passo 6 e generare il risultato finale.

Va notato che l'OR esclusivo con ipad porta a complementare una metà dei bit di K . In modo analogo, l'OR esclusivo con opad porta a complementare un'altra metà dei bit di K , contenente bit diversi. In effetti, sottponendo S_i e S_o all'algoritmo hash, si generano, in maniera pseudo-casuale, due chiavi a partire da K .

HMAC richiede, in generale, un tempo di esecuzione approssimativamente uguale a quello della funzione hash in esso contenuta nel caso di messaggi lunghi. In particolare, HMAC aggiunge tre esecuzioni a quelle della funzione hash di base (quelle per S_i , S_o , e per il blocco prodotto dall'hash interno).

3.3 Principi di crittografia a chiave pubblica

La crittografia a chiave pubblica, di uguale importanza rispetto a quella convenzionale, trova applicazione per scopi di autenticazione dei messaggi e distribuzione di chiavi. In questo paragrafo sono dapprima esaminati i concetti base della crittografia a chiave pubblica e, successivamente, sarà fornito un quadro introduttivo delle problematiche di distribuzione delle chiavi. Nel Paragrafo 3.4 saranno considerati i due principali algoritmi a chiave pubblica: RSA e Diffie-Hellman. Il Paragrafo 3.5 introdurrà, infine, le firme digitali.

Struttura della cifratura a chiave pubblica

La cifratura a chiave pubblica, proposta per la prima volta da Diffie e Hellman nel 1976 [DIFF76], rappresenta il primo, rivoluzionario passo avanti compiuto nel campo della ci-

fratura da migliaia di anni. Gli algoritmi a chiave pubblica si basano su funzioni matematiche piuttosto che su semplici operazioni su sequenze di bit, come quelle che vengono usate negli algoritmi di cifratura simmetrici. Ma soprattutto, contrariamente alla cifratura simmetrica convenzionale che impiega una sola chiave, la crittografia a chiave pubblica è asimmetrica e fa uso di due chiavi distinte. L'uso di due chiavi comporta profonde conseguenze per gli aspetti di riservatezza, distribuzione delle chiavi ed autenticazione.

Prima di proseguire con la trattazione, è opportuno soffermarsi su alcuni diffusi preconcetti inerenti la crittografia a chiave pubblica. Il primo sostiene che la cifratura a chiave pubblica è più sicura di quella convenzionale rispetto alla crittoanalisi. In effetti, la sicurezza di uno schema di cifratura dipende (1) dalla lunghezza della chiave e (2) dallo sforzo computazionale richiesto per rendere inefficace il cifrario. In linea di principio, non esiste nulla che renda una tipologia di cifratura migliore dell'altra dal punto di vista della resistenza agli attacchi di crittoanalisi. Una seconda concezione errata afferma che la cifratura a chiave pubblica è una tecnica di uso generale (*general purpose*) che ha reso obsoleta la cifratura convenzionale. Al contrario, non sembra probabile che la cifratura convenzionale venga abbandonata a causa del sovraccarico di elaborazione degli attuali schemi di cifratura a chiave pubblica. Infine, la distribuzione delle chiavi nella cifratura a chiave pubblica appare banale, soprattutto se confrontata con la complessità delle procedure di handshaking connesse con i centri di distribuzione di chiavi nel caso di cifratura convenzionale. In pratica, è necessario avere qualche forma di protocollo, sovente facendo uso di un agente centrale, e le procedure impiegate non sono né più semplici né più efficienti di quelle richieste dalla cifratura convenzionale.

Uno schema a chiave pubblica presenta sei componenti (Figura 3.7a).

- **Testo in chiaro:** il messaggio o i dati in formato leggibile forniti in ingresso all'algoritmo.
- **Algoritmo di cifratura:** esegue un certo numero di trasformazioni sul testo in chiaro.
- **Chiave pubblica e chiave privata:** coppia di chiavi scelte in modo che una venga utilizzata per la cifratura e l'altra per la decifratura. Le trasformazioni effettuate dall'algoritmo di cifratura dipendono dalla chiave pubblica o privata fornita in ingresso.
- **Testo cifrato:** messaggio prodotto come risultato della cifratura. Il testo cifrato dipende dal testo in chiaro e dalla chiave. Dato un messaggio, due chiavi diverse produrranno due diversi testi cifrati.
- **Algoritmo di decifratura:** questo algoritmo riceve il testo cifrato e la corrispondente chiave e produce il testo in chiaro originario.

Come suggeriscono i nomi, la chiave pubblica della coppia di chiavi è nota a tutti, mentre la chiave privata è conosciuta solo dal suo proprietario. Un algoritmo di uso generale di cifratura a chiave pubblica presuppone una chiave per la fase di cifratura e una chiave diversa ma correlata alla precedente per la fase di decifratura. I passi fondamentali sono i seguenti.

1. Ogni utente genera una coppia di chiavi da usare per la cifratura e decifratura dei messaggi.
2. Ciascun utente pone una delle due chiavi in un registro pubblico o in qualche altro file accessibile pubblicamente. Questa è la chiave pubblica. L'altra chiave della coppia

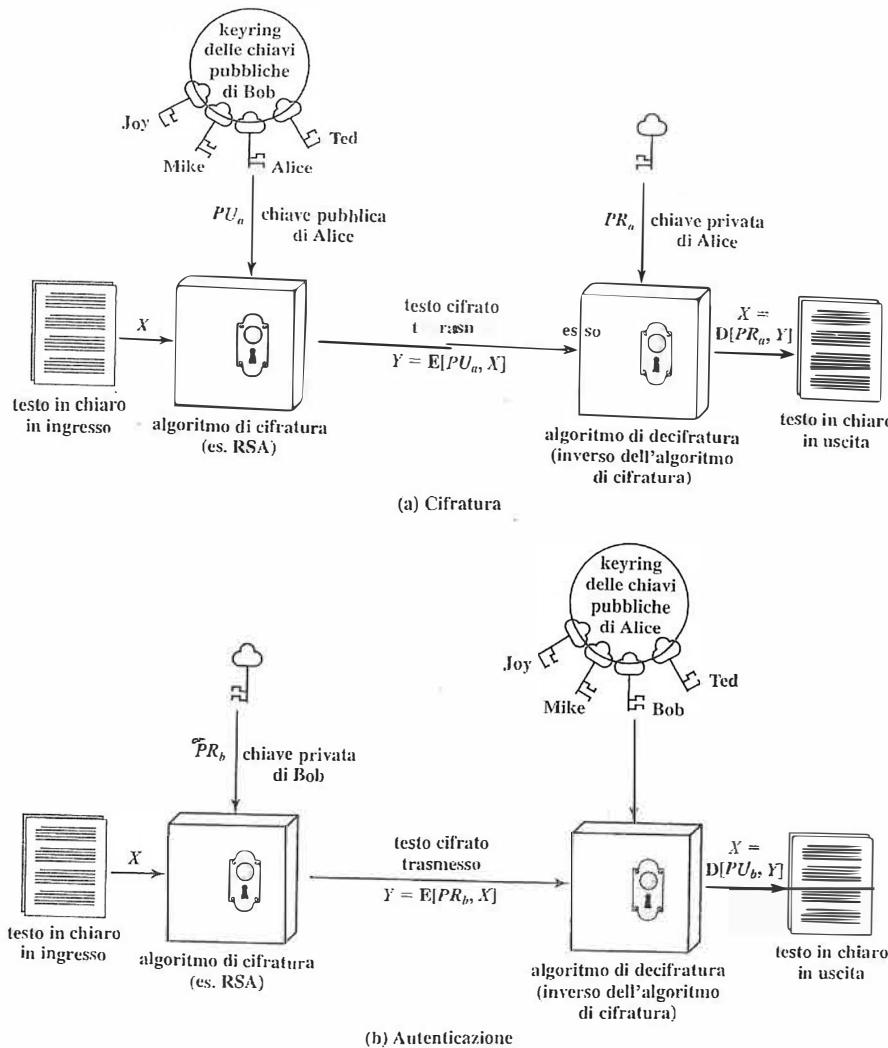


Figura 3.7 Crittografia a chiave pubblica.

è mantenuta privata. Come mostrato nella Figura 3.7a, ciascun utente mantiene un insieme di chiavi pubbliche ricevute da altri utenti.

- Se Bob vuole inviare un messaggio privato ad Alice, cifra il messaggio utilizzando la chiave pubblica di Alice.

- Quando Alice riceve il messaggio, lo decifra utilizzando la propria chiave privata. Nessun altro utente può decifrare il messaggio in quanto solo Alice conosce la propria chiave segreta.

In base a questo approccio, tutti i partecipanti hanno accesso alle chiavi pubbliche, mentre le chiavi private vengono generate localmente da ciascun partecipante e, quindi, non esiste alcuna necessità di distribuirle. Fino a che un utente mantiene protetta la propria chiave privata, le comunicazioni a lui dirette sono sicure. In qualunque momento, un utente può cambiare la propria chiave privata e rendere nota la corrispondente nuova chiave pubblica che sostituisce la vecchia. La chiave usata nella cifratura convenzionale viene detta chiave segreta, mentre le due chiavi usate nella cifratura a chiave pubblica sono note rispettivamente come chiave pubblica e chiave privata. La chiave privata viene sempre mantenuta segreta, ma viene chiamata chiave privata e non chiave segreta per non fare confusione con la cifratura convenzionale.

Applicazioni dei sistemi di crittografia a chiave pubblica

Prima di proseguire, occorre chiarire un aspetto relativo ai sistemi di crittografia a chiave pubblica che potrebbe altrimenti generare confusione. Questi sistemi sono caratterizzati dall'impiego di un algoritmo di cifratura con due chiavi, una mantenuta privata e una disponibile pubblicamente. A seconda dell'applicazione, il mittente utilizza la propria chiave privata o la chiave pubblica del destinatario o entrambe, al fine di eseguire qualche tipologia di funzione crittografica. A grandi linee è possibile classificare l'uso dei sistemi di crittografia a chiave pubblica in tre categorie.

- Cifratura/decifratura:** il mittente cifra un messaggio con la chiave pubblica del ricevente.
- Firma digitale:** il mittente "firma" un messaggio con la propria chiave privata. La firma si ottiene attraverso un algoritmo di cifratura applicato al messaggio o a un piccolo blocco di dati che è funzione del messaggio.
- Scambio di chiavi:** due parti collaborano al fine di scambiarsi una chiave di sessione. Sono possibili differenti approcci che fanno uso della/le chiave/i privata/e di una o di entrambe le parti.

Alcuni algoritmi sono adatti per tutte le tre applicazioni, mentre altri possono essere usati solamente per una o due di queste. La Tabella 3.2 indica le applicazioni possibili per gli algoritmi RSA e Diffie-Hellman, e per la firma digitale standard (DSS) e la crittografia delle curve ellittiche. Questi argomenti saranno approfonditi successivamente in questo capitolo.

Algoritmo	Cifratura/Decifratura	Firma Digitale	Scambio di chiavi
RSA	sì	sì	sì
Diffie-Hellman	no	no	sì
DSS	no	sì	no
Curve Ellittiche	sì	sì	sì

Tabella 3.2 Applicazioni dei sistemi di crittografia a chiave pubblica.

Requisiti della crittografia a chiave pubblica

Il sistema di crittografia della Figura 3.7 dipende da un algoritmo di crittografia basato su due chiavi correlate. Diffie e Hellman hanno ipotizzato questo sistema senza dimostrare l'esistenza di tale algoritmo. Hanno comunque stabilito le condizioni che l'algoritmo deve soddisfare [DIFF76].

1. È computazionalmente facile per una parte B generare una coppia (chiave pubblica PU_b , chiave privata PR_b).
2. È computazionalmente facile per un mittente A, a conoscenza della chiave pubblica e del messaggio da cifrare, M , generare il testo cifrato corrispondente:

$$C = E(PU_b, M)$$

3. È computazionalmente facile per il ricevente B decifrare il testo cifrato risultante utilizzando la chiave privata per ricostruire il messaggio originale:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. È computazionalmente improponibile per un avversario, a conoscenza della chiave pubblica, PU_b , determinare la chiave privata corrispondente, PR_b .
5. È computazionalmente improponibile per un avversario, a conoscenza della chiave pubblica, PU_b , e del testo cifrato, C , ricostruire il messaggio originario M .

È possibile aggiungere un sesto requisito ai precedenti che, pur essendo utile, non è necessario per tutte le applicazioni a chiave pubblica.

6. Ciascuna delle due chiavi correlate può essere utilizzata per la cifratura, usando l'altra per la decifratura.

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

3.4 Algoritmi di crittografia a chiave pubblica

I due algoritmi a chiave pubblica usati più diffusamente sono l'algoritmo RSA e quello di Diffie-Hellman, qui di seguito trattati. Successivamente, verranno presentati sinteticamente altri due algoritmi⁴.

Algoritmo a chiave pubblica RSA

Uno dei primi schemi a chiave pubblica fu sviluppato nel 1977 da Ron Rivest, Adi Shamir, e Len Adleman al MIT e pubblicato per la prima volta nel 1978 [RIVE78]. Da allora, lo schema RSA ha regnato sovrano come l'approccio per la cifratura a chiave pubblica maggiormente accettato e utilizzato nelle implementazioni. RSA è un cifrario a blocchi in cui il testo in chiaro e quello cifrato sono interi compresi fra 0 e $n - 1$, dato un valore n .

Dato un blocco di testo in chiaro M e un blocco di testo cifrato C , la cifratura e la decifratura sono definite come segue:

$$C = M^e \text{ mod } n$$

$$M = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$$

Sia il mittente sia il ricevente devono conoscere i valori di n e di e , mentre solo il ricevente conosce il valore di d . Questo è un algoritmo di cifratura a chiave pubblica con chiave pubblica $KU = \{e, n\}$ e chiave privata $KR = \{d, n\}$. Affinché questo algoritmo possa essere utilizzato proficuamente per la cifratura a chiave pubblica, devono essere verificati i seguenti requisiti.

1. È possibile trovare valori di e, d, n tali che $M^{ed} = M \text{ mod } n$ per ogni $M < n$.
2. È relativamente semplice calcolare M^e e C^d per tutti i valori di $M < n$.
3. Dati e ed n , è improponibile determinare d .

I primi due requisiti si ottengono facilmente. Il terzo può essere ottenuto considerando valori di e e di n molto grandi.

L'algoritmo RSA è sintetizzato nella Figura 3.8. L'algoritmo inizia selezionando due numeri primi, p e q , e calcolando il loro prodotto n , che diventa il modulo per la cifratura e decifratura. Successivamente, si determina la quantità $\phi(n)$, nota come funzione di Eulero di n , pari al numero di interi positivi minori di n e relativamente primi rispetto a n . Viene quindi selezionato un intero e , primo rispetto a $\phi(n)$, cioè tale che il massimo comun divisore fra e e $\phi(n)$ sia 1. Infine, l'algoritmo calcola d come l'inverso moltiplicativo di e , modulo $\phi(n)$. Si può dimostrare che d ed e possiedono le proprietà desiderate.

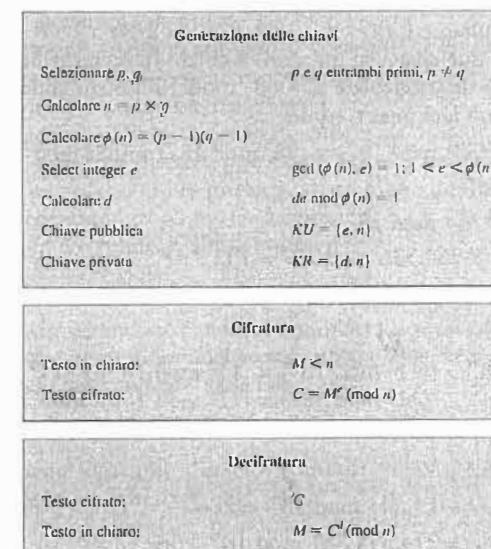


Figura 3.8 Algoritmo RSA.

⁴ Questo paragrafo usa alcuni concetti elementari della teoria dei numeri. Per un ripasso si veda l'Appendice A.

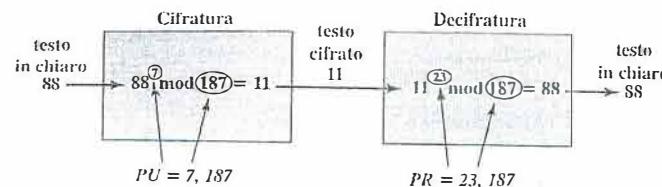


Figura 3.9 Esempio di applicazione dell'algoritmo RSA.

Si supponga che l'utente A abbia reso nota la sua chiave pubblica e che l'utente B voglia inviargli un messaggio M . Allora B calcola $C = M^e \pmod{n}$ e trasmette C . Alla ricezione di questo testo cifrato, l'utente A lo decifra calcolando $M = C^d \pmod{n}$.

La Figura 3.9 riporta un esempio tratto da [SING99] in cui le chiavi sono state generate nel seguente modo.

1. Selezione di due numeri primi $p = 17$ e $q = 11$.
2. Calcolo di $n = pq = 17 \times 11 = 187$.
3. Calcolo di $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Selezione di e tale che e sia relativamente primo rispetto a $\phi(n) = 160$ e minore di $\phi(n)$; in questo caso $e = 7$.
5. Determinare d tale che $de \equiv 1 \pmod{160}$ e $d < 160$. Il valore corretto è $d = 23$, poiché $23 \times 7 = 161 = 10 \times 160 + 1$.

La chiave pubblica risultante è $PU = \{7, 187\}$ e quella privata è $PR = \{23, 187\}$. L'esempio mostra l'uso di queste due chiavi su un testo in chiaro in ingresso $M = 88$.

Per la cripfratura si deve calcolare $C = 88^7 \pmod{187}$. Sfruttando le proprietà dell'aritmetica modulare, si può fare quanto segue:

$$\begin{aligned} 88^7 \pmod{187} &= [(88^4 \pmod{187}) \times (88^2 \pmod{187}) \times (88^1 \pmod{187})] \pmod{187} \\ 88^1 \pmod{187} &= 88 \\ 88^2 \pmod{187} &= 7744 \pmod{187} = 77 \\ 88^4 \pmod{187} &= 59.969.536 \pmod{187} = 132 \\ 88^7 \pmod{187} &= (88 \times 77 \times 132) \pmod{187} = 894.432 \pmod{187} = 11 \end{aligned}$$

Per la decifratura si calcola $M = 11^{23} \pmod{187}$:

$$\begin{aligned} 11^{23} \pmod{187} &= [(11^1 \pmod{187}) \times (11^2 \pmod{187}) \times (11^4 \pmod{187}) \times \\ &\quad (11^8 \pmod{187}) \times (11^8 \pmod{187})] \pmod{187} \\ 11^1 \pmod{187} &= 11 \\ 11^2 \pmod{187} &= 121 \\ 11^4 \pmod{187} &= 14.641 \pmod{187} = 55 \\ 11^8 \pmod{187} &= 214.358.881 \pmod{187} = 33 \\ 11^{23} \pmod{187} &= (11 \times 121 \times 55 \times 33 \times 33) \pmod{187} \\ &= 79.720.245 \pmod{187} = 88 \end{aligned}$$

Esistono due possibili approcci per rendere inefficace l'algoritmo RSA. Il primo è l'approccio a forza bruta, ossia provare tutte le possibili chiavi private: quindi, tanto più grande è il numero di bit in e e in d , tanto più sicuro è l'algoritmo. Tuttavia, dato che i calcoli svolti – sia nella generazione delle chiavi sia durante la cripfratura e decifratura – sono complessi, quanto maggiore è la dimensione della chiave tanto più lenta diventa l'esecuzione.

La maggior parte delle discussioni circa la crittoanalisi dell'algoritmo RSA si è concentrata sul compito di scomporre n nei suoi due fattori primi. Per valori grandi di n con fattori primi grandi, la scomposizione in fattori primi è un problema molto complesso, ma non tanto quanto si crede. Quella che segue è un'impressionante dimostrazione di questo fatto. Nel 1977, i tre inventori dell'algoritmo RSA sfidaroni i lettori di *Scientific American* a decodificare il testo cifrato pubblicato nella colonna "Mathematical Games" di Martin Gardner [GARD77]. Misero in palio un compenso di 100 dollari per chi avesse fatto pervenire il corrispondente testo in chiaro, evento che predissero non si sarebbe verificato per 40 quadrillioni di anni circa. Nell'aprile del 1994, un gruppo operante su Internet con oltre 1600 calcolatori chiese il compenso dopo soli otto mesi di lavoro [LEUT94]. Per questa sfida venne utilizzata una dimensione di chiave pubblica (lunghezza di n) di 129 cifre decimali, cioè circa 428 bit. Questo risultato non vanifica l'impiego di RSA; dimostra semplicemente che devono essere impiegate chiavi di maggiore lunghezza. Attualmente, si considera una dimensione di chiave di 1024 bit (circa 300 cifre decimali) abbastanza robusta per tutte le applicazioni.

Algoritmo Diffie-Hellman per lo scambio di chiavi

Il primo algoritmo a chiave pubblica apparve nell'articolo originario di Diffie e Hellman, in cui si definiva la crittografia a chiave pubblica [DIFF76], ed è generalmente conosciuto come algoritmo Diffie-Hellman per lo scambio di chiavi. Molti prodotti commerciali utilizzano questa tecnica di scambio di chiavi.

Scopo dell'algoritmo è quello di consentire a due utenti di scambiarsi in maniera sicura una chiave segreta che può poi essere utilizzata per la successiva cripfratura di messaggi. L'algoritmo stesso si limita allo scambio delle chiavi.

L'efficacia dell'algoritmo Diffie-Hellman dipende dalla difficoltà di calcolare logaritmi discreti. In breve, un logaritmo discreto può essere definito nel seguente modo. Prima di tutto, va introdotta la nozione di radice primitiva di un numero primo p , intesa come il numero le cui potenze generano tutti gli interi compresi fra 1 e $p - 1$. In altre parole, se a è la radice primitiva del numero primo p , allora i numeri

$$a \pmod{p}, a^2 \pmod{p}, \dots, a^{p-1} \pmod{p}$$

sono numeri distinti e corrispondono agli interi da 1 fino a $p - 1$ permutati in qualche modo.

Dato un qualunque numero intero b e una radice primitiva a di un numero primo p , è possibile trovare un unico esponente i tale che

$$b = a^i \pmod{p} \text{ con } 0 \leq i \leq (p - 1)$$

L'esponente i è chiamato logaritmo discreto, o indice⁵, di b rispetto alla base $a \pmod{p}$. Questo valore viene denotato con $\text{dlog}_{a,p}(b)$.

⁵ Molti testi fanno riferimento al logaritmo discreto come all'*indice*. Non c'è un accordo generale sulla notazione di questo concetto, tanto meno un nome concordato.

L'algoritmo

Sfruttando queste nozioni di base, si procede ora a descrivere l'algoritmo Diffie-Hellman per lo scambio di chiavi, schematizzato nella Figura 3.10. In questo schema, esistono due numeri noti pubblicamente: un numero primo q e un intero α primitiva di q . Si supponga che gli utenti A e B vogliano scambiarsi una chiave. L'utente A sceglie un intero casuale $X_A < q$ e calcola $Y_A = \alpha^{X_A} \bmod q$. In maniera analoga, l'utente B seleziona indipendentemente un intero casuale $X_B < q$ e calcola $Y_B = \alpha^{X_B} \bmod q$. Ciascuna delle parti mantiene il valore X privato e rende il valore Y disponibile pubblicamente all'altra parte. L'utente A calcola la chiave secondo lo schema $K = (Y_B)^{X_A} \bmod q$ e l'utente B calcola la chiave come $K = (Y_A)^{X_B} \bmod q$. Questi due calcoli producono risultati identici:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

Il risultato è che le due parti si sono scambiate un valore segreto. Inoltre, poiché X_A e X_B sono private, l'avversario dispone solamente dei seguenti elementi con cui operare: q , α , Y_A e Y_B . Quindi, per determinare la chiave l'avversario è costretto a calcolare un logaritmo discreto.

Ad esempio, per attaccare la chiave segreta dell'utente B, l'avversario deve calcolare

$$X_B = \text{dlog}_{\alpha, q}(Y_B)$$

L'avversario può quindi calcolare la chiave K in maniera analoga a come viene calcolata da B.

La sicurezza dell'algoritmo Diffie-Hellman per lo scambio di chiavi sfrutta il fatto che mentre è relativamente facile calcolare potenze modulo un numero primo, è molto complesso calcolare logaritmi discreti. Per numeri primi grandi, quest'ultimo compito è considerato improponibile.

Di seguito riportiamo un esempio. Lo scambio di chiavi si basa sull'impiego del numero primo $q = 353$ e di una radice primitiva di 353, in questo caso $\alpha = 3$. A e B scelgono rispettivamente le chiavi private $X_A = 97$ e $X_B = 233$. Ciascun utente calcola la propria chiave pubblica:

$$\text{A calcola } Y_A = 3^{97} \bmod 353 = 40$$

$$\text{B calcola } Y_B = 3^{233} \bmod 353 = 248$$

Dopo essersi scambiati le rispettive chiavi pubbliche, ciascuno dei due può calcolare la chiave segreta comune:

$$\text{A calcola } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$

$$\text{B calcola } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

Si ipotizzi che un attaccante abbia disponibili le seguenti informazioni:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

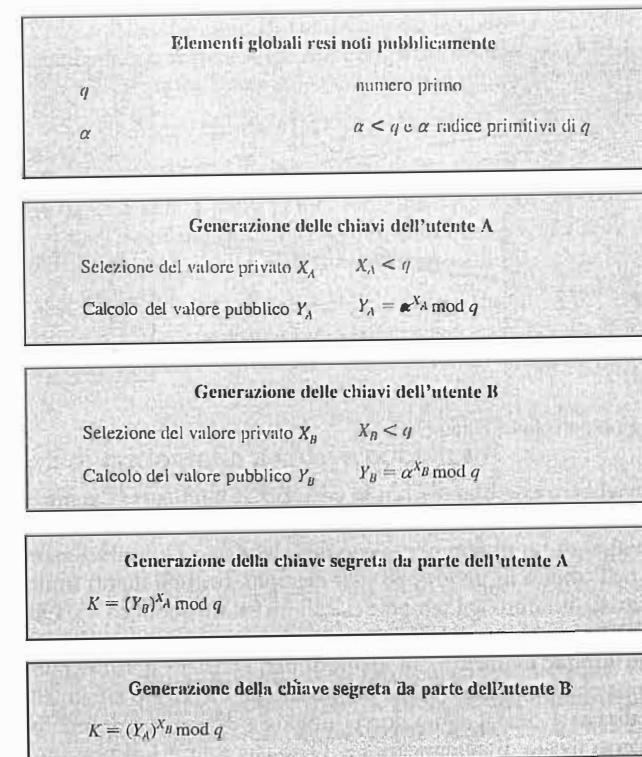


Figura 3.10 Algoritmo Diffie-Hellman per lo scambio di chiavi.

In questo semplice esempio, sarebbe possibile con l'approccio a forza bruta, determinare la chiave segreta 160. In particolare, un attaccante E può determinare la chiave comune, scoprindo una soluzione dell'equazione $3^a \bmod 353 = 40$ oppure $3^b \bmod 353 = 248$. L'approccio a forza bruta consiste nel calcolare le potenze di 3 mod 353, fermandosi quando il risultato è 40 o 248. La risposta desiderata viene raggiunta con il valore dell'esponente di 97, che fornisce $3^{97} \bmod 353 = 40$.

Con numeri più grandi il problema diventa irrisolvibile.

Protocolli per lo scambio delle chiavi

Nella Figura 3.11 è riportato un semplice protocollo che utilizza lo schema di calcolo di Diffie-Hellman. Si supponga che A voglia instaurare una connessione con l'utente B, utilizzando una chiave segreta per cifrare i messaggi su quella connessione. L'utente A può generare una chiave privata X_A da usare una sola volta, calcolare Y_A e inviarlo all'utente B. A sua volta l'utente B risponde generando un valore privato X_B , calcolando Y_B e mandandolo all'utente A. Entrambi gli utenti possono a questo punto calcolare la chiave segreta. I valori pubblici ne-

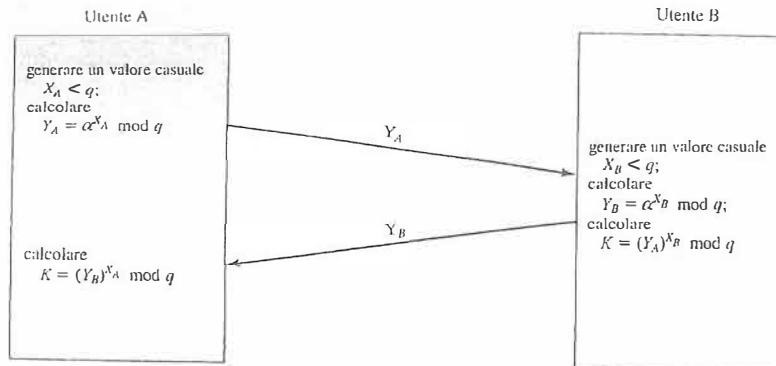


Figura 3.11 Scambio di chiavi Diffie-Hellman.

cessari q e α dovrebbero essere conosciuti in anticipo; in alternativa l'utente A potrebbe scegliere i valori q e α e includerli nel primo messaggio inviato.

Come esempio di un ulteriore impiego dell'algoritmo Diffie-Hellman, si ipotizzi che ciascun utente nell'ambito di un gruppo (per esempio, tutti gli utenti su una LAN) generi un valore privato X_A duraturo nel tempo e calcoli un valore pubblico Y_A . Questi valori pubblici, assieme ai valori pubblici globali q e α , sono memorizzati in una directory centralizzata. In un qualunque momento, un utente B può accedere al valore pubblico dell'utente A, calcolare una chiave segreta e usarla per inviare un messaggio cifrato all'utente A. Se la directory centralizzata è fidata, allora questa tipologia di comunicazione fornisce sia riservatezza sia un certo livello di autenticazione. Dato che solo A e B possono determinare la chiave, nessun altro utente può leggere il messaggio (riservatezza). Il ricevente A è a conoscenza che solo l'utente B avrebbe potuto creare un messaggio usando questa chiave (autenticazione). In ogni caso, questa tecnica non protegge dagli attacchi di replay.

Attacchi Man-in-the-middle

Il protocollo rappresentato nella Figura 3.11 è insicuro contro gli attacchi *man-in-the-middle* (uomo nel mezzo). Si supponga che Alice e Bob vogliano scambiarci le chiavi e che Darth sia l'avversario. L'attacco procede nel seguente modo.

1. Darth si prepara all'attacco generando due chiavi private casuali X_{D1} e X_{D2} e poi calcola le corrispondenti chiavi pubbliche Y_{D1} e Y_{D2} .
2. Alice trasmette Y_A a Bob.
3. Darth intercetta Y_A e trasmette Y_{D1} a Bob. Darth calcola anche $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob riceve Y_{D1} e calcola $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob trasmette Y_B ad Alice.
6. Darth intercetta Y_B e trasmette Y_{D2} a Alice. Darth calcola anche $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. Alice riceve Y_{D2} e calcola $K1 = (Y_{D2})^{X_A} \bmod q$.

A questo punto Bob e Alice pensano di condividere una chiave segreta, ma invece Bob e Darth condividono la chiave segreta $K1$ e Alice e Darth condividono la chiave segreta $K2$. Tutte le future comunicazioni tra Bob e Alice saranno compromesse nel seguente modo.

1. Alice manda un messaggio cifrato $M: E(K3, M)$.
2. Darth intercetta il messaggio cifrato e lo decifra per ricavare M .
3. Darth manda a Bob $E(K1, M)$ o $E(K1, M')$, dove M' è un qualsiasi messaggio. Nel primo caso, Darth vuole intercettare la comunicazione senza modificarla. Nel secondo caso Darth vuole modificare il messaggio indirizzato a Bob.

Il protocollo di scambio delle chiavi è vulnerabile a questo tipo di attacchi perché non autentica i partecipanti. Questa vulnerabilità può essere superata con l'uso della firma digitale e di certificati a chiave pubblica. Questi argomenti verranno spiegati più avanti in questo capitolo e nel Capitolo 4.

Altri algoritmi di crittografia a chiave pubblica

Altri due algoritmi a chiave pubblica hanno trovato consenso in ambito commerciale: l'algoritmo DSS (*digital signature standard*) e la crittografia delle curve ellittiche (ECC, *elliptic-curve cryptography*).

Algoritmo DSS

Il National Institute of Standards and Technology (NIST) ha pubblicato il documento Federal Information Processing Standard FIPS PUB 186, conosciuto come Digital signature standard (DSS) che fa uso dell'algoritmo SHA-1 e propone una nuova tecnica di firma digitale, l'algoritmo di firma digitale (DSA, *digital signature algorithm*). DSS fu originariamente proposto nel 1991 e revisionato nel 1993 per rispondere a feedback relativi alla sicurezza dello schema. Ci fu un'ulteriore revisione minore nel 1996. DSS utilizza un algoritmo progettato per fornire solamente la funzione di firma digitale. A differenza di RSA, non può essere impiegato per scopi di cifratura o di scambio di chiavi.

Crittografia delle curve ellittiche

La maggior parte dei prodotti e degli standard che utilizzano la crittografia a chiave pubblica per cifratura e firma digitale utilizzano dell'algoritmo RSA. Per un impiego sicuro di RSA, la lunghezza in bit della chiave è aumentata negli ultimi anni, rendendo più oneroso il carico di elaborazione delle applicazioni basate su RSA. Questo carico aggiuntivo ha conseguenze, specialmente per i siti di commercio elettronico che eseguono un numero elevato di transazioni sicure. Recentemente, un sistema ha iniziato a sfidare RSA: la crittografia delle curve ellittiche (ECC, *elliptic-curve cryptography*). ECC si sta facendo notare dal punto di vista degli sforzi di standardizzazione, comprendendo lo standard IEEE P1363 per la crittografia a chiave pubblica.

La principale attrattiva di ECC rispetto a RSA consiste nel fatto che sembra fornire lo stesso livello di sicurezza con una dimensione inferiore di bit, riducendo pertanto il sovraccarico di elaborazione. D'altro canto, anche se la teoria su cui si basa ECC è nota da diverso tempo, solo recentemente sono apparsi i primi prodotti e si è manifestato un notevole in-

teresse verso la crittoanalisi per dimostrarne i punti deboli. Il livello di fiducia per ECC non è pertanto così elevato come per RSA.

ECC è sostanzialmente più difficile da spiegare rispetto a RSA o a Diffie-Hellman, e una descrizione matematica completa va oltre gli scopi di questo testo. La tecnica si basa sull'impiego del costrutto matematico noto come curva ellittica.

3.5 Firme digitali

La cifratura a chiave pubblica può essere usata in un altro modo, come illustrato nella Figura 3.7b. Si supponga che Bob voglia inviare un messaggio ad Alice e, anche se non è importante che il messaggio sia tenuto segreto, desidera che Alice sia certa che il messaggio provenga proprio da lui. In questo caso, Bob usa la propria chiave privata per cifrare il messaggio. Quando Alice riceve il testo cifrato, scopre di poterlo decifrare con la chiave pubblica di Bob, potendo pertanto provare che il messaggio è stato cifrato da lui. Nessun altro possiede la chiave privata di Bob e perciò nessun altro avrebbe potuto creare un testo cifrato decifrabile mediante la chiave pubblica di Bob. Di conseguenza, l'intero messaggio cifrato ha il ruolo di firma digitale. Inoltre, è impossibile modificare il messaggio senza accedere alla chiave privata di Bob e, quindi, il messaggio risulta autenticato sia rispetto alla sorgente sia rispetto all'integrità dei dati contenuti.

Nello schema precedente, se da un lato la cifratura dell'intero messaggio consente di validare tanto l'autore che il contenuto, dall'altro lato richiede grandi capacità di memorizzazione. Ciascun documento deve essere mantenuto in chiaro per poter essere usato ai fini pratici cui è destinato. Inoltre, occorre memorizzare una copia cifrata del documento in modo da poter verificare la sorgente e i contenuti in caso di controversie. Un modo più efficiente per ottenere i medesimi risultati consiste nel cifrare un piccolo blocco di bit, in funzione del documento. Tale blocco, denominato autenticatore, deve soddisfare la proprietà per cui risulta improponibile alterare un documento senza che venga modificato l'autenticatore ad esso associato. Se l'autenticatore è cifrato mediante la chiave privata del mittente, serve da firma per la verifica della sorgente, del contenuto e della sequenzializzazione. A tale scopo si può utilizzare un codice hash sicuro come, ad esempio, SHA-1. Questo scenario è mostrato nella Figura 3.2b.

È importante sottolineare che il processo di cifratura appena descritto non fornisce riservatezza. Vale a dire, il messaggio da trasmettere è protetto da modifiche ma non da intercettazioni. Questo è ovvio nel caso di firma basata su una porzione del messaggio, perché il resto del messaggio viene trasmesso in chiaro. Ma anche nel caso di cifratura completa non esiste protezione rispetto alla riservatezza, in quanto un qualunque osservatore potrebbe decifrare il messaggio utilizzando la chiave pubblica del mittente.

3.6 Gestione delle chiavi

Uno fra i principali ruoli della cifratura a chiave pubblica è quello relativo alla distribuzione delle chiavi. Rispetto a ciò, si distinguono due diversi aspetti circa l'uso della cifratura a chiave pubblica:

- distribuzione di chiavi pubbliche;
- uso di cifratura a chiave pubblica per la distribuzione di chiavi segrete.

Certificati a chiave pubblica

Il punto fondamentale della cifratura a chiave pubblica è che questa, come dice il nome, è pubblica. Pertanto, con un algoritmo a chiave pubblica sufficientemente diffuso, come ad esempio RSA, un qualunque partecipante può inviare la propria chiave pubblica a un qualunque altro partecipante, oppure inviarla in modalità broadcast a una comunità di utenti. Pur essendo conveniente, questo approccio ha una debolezza non trascurabile: chiunque potrebbe contraffare tale informazione pubblica. Ovvero, qualche utente potrebbe pretendere di essere l'utente A e inviare una chiave pubblica a un altro partecipante, oppure inviarla in modalità broadcast. Fintantoché l'utente A non scopre la manomissione e non mette in guardia gli altri partecipanti, l'avversario è in grado di leggere tutti i messaggi destinati ad A e di utilizzare le chiavi contraffatte per effettuare autenticazione.

La soluzione a questo problema è rappresentata dai certificati a chiave pubblica. In sintesi, un certificato consiste in una chiave pubblica e in un ID utente del possessore della chiave, con la firma di una terza parte fidata apposta su tutto il blocco. Solitamente, la terza parte è un'autorità certificativa (CA, *certificate authority*) che viene ritenuta fidata dalla comunità degli utenti come, ad esempio, un'agenzia governativa oppure un'istituzione finanziaria. Un utente può presentare la propria chiave pubblica all'autorità in maniera sicura e ottenere un certificato. L'utente può quindi pubblicare il certificato. Chiunque avesse bisogno della chiave pubblica di questo utente può ottenere il certificato e verificarne la validità attraverso la firma fidata annessa. Il processo è mostrato nella Figura 3.12.

Lo standard X.509 è stato universalmente accettato per la creazione di certificati a chiave pubblica. I certificati X.509 sono usati nella maggior parte delle applicazioni di sicurezza di rete, compresi: la sicurezza IP, SSL, le transazioni elettroniche sicure (SET) e S/MIME. Questi argomenti saranno affrontati nella seconda parte del libro, mentre X.509 è esaminato in dettaglio nel Capitolo 4.

Distribuzione a chiave pubblica di chiavi segrete

Con la cifratura convenzionale, requisito fondamentale affinché due parti possano comunicare in maniera sicura è la condivisione di una chiave segreta. Si supponga che Bob voglia creare un'applicazione per la trasmissione di messaggi, che gli consenta di scambiare posta elettronica in maniera sicura con chiunque abbia accesso a Internet o a qualunque altra rete di comunicazione utilizzabile a tale scopo. Inoltre, si assuma che Bob desideri operare utilizzando la cifratura convenzionale. Con la cifratura convenzionale Bob e l'utente con cui avviene lo scambio di posta elettronica, ad esempio Alice, devono trovare un modo per condividere una chiave segreta unica, di cui nessun altro sia a conoscenza.

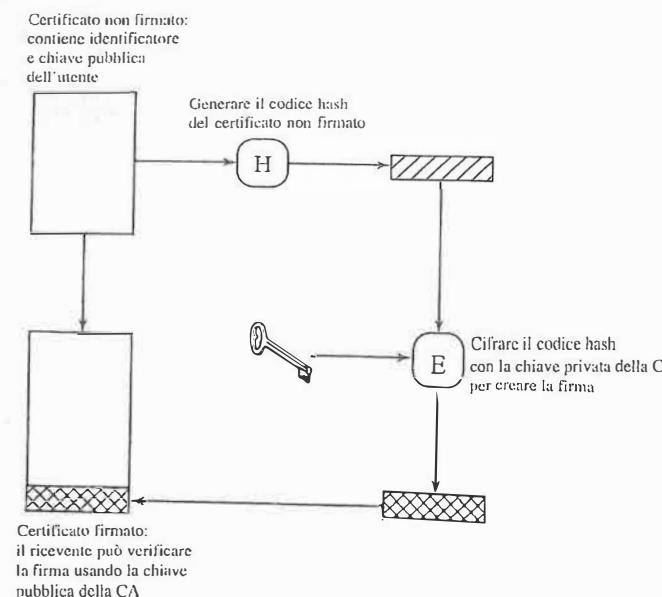


Figura 3.12 Uso dei certificati a chiave pubblica.

Come si può fare? Se Alice si trova nella stanza attigua a quella di Bob, Bob potrebbe generare una chiave e scriverla su un pezzo di carta o memorizzarla su un dischetto e consegnarla manualmente ad Alice. Ma se Alice si trova dall'altra parte del continente o del pianeta, come si può procedere? Bob potrebbe cifrare la chiave mediante cifratura convenzionale e inviarla mediante posta elettronica ad Alice, ma questo implica che entrambi siano già a conoscenza di una chiave segreta con cui cifrare questa seconda chiave segreta. Inoltre, Bob, e chiunque altro faccia uso di questo nuovo servizio di posta elettronica, dovrebbe trattare questo stesso problema con ciascun potenziale corrispondente: tutte le coppie di utenti che desiderano scambiarsi messaggi di posta elettronica devono condividere un'unica chiave segreta.

Un possibile approccio consiste nell'utilizzare l'algoritmo Diffie-Hellman per lo scambio di chiavi. Questo approccio risulta infatti ampiamente utilizzato. Tuttavia, nella sua forma più semplice, l'approccio presenta un inconveniente: non esiste alcuna autenticazione delle due parti in comunicazione.

Un potente approccio alternativo consiste nell'utilizzare certificati a chiave pubblica. Quando Bob vuole comunicare con Alice, può procedere nel seguente modo:

1. preparare un messaggio;
2. cifrare il messaggio con tecniche di cifratura convenzionale, attraverso una chiave di sessione utilizzata una sola volta;

3. cifrare la chiave di sessione mediante cifratura a chiave pubblica utilizzando la chiave pubblica di Alice;

4. accludere la chiave di sessione cifrata al messaggio e inviare il tutto ad Alice.

Solo Alice è in grado di decifrare la chiave di sessione e, quindi, ricostruire il messaggio originario. Se Bob ha ottenuto la chiave pubblica di Alice, attraverso il certificato a chiave pubblica di Alice, ha garanzia che sia una chiave valida.

3.7 Letture consigliate e siti web

Validi trattamenti delle funzioni hash e dei codici di autenticazione dei messaggi si trovano in [STIN06] e [MENE97].

Le trattazioni consigliate per la cifratura fornite al Capitolo 2 comprendono sia i concetti di chiave pubblica sia di cifratura tradizionale. [DIFF88] descrive in dettaglio i molti tentativi di escogitare critto-algoritmi sicuri a due chiavi e la graduale evoluzione di una varietà di protocolli basati su di essi. [CORM01] fornisce un riassunto conciso, ma completo e di agevole lettura, di tutti i più importanti algoritmi relativi alla verifica, al calcolo e alla critto-analisi di RSA.

Siti web consigliati

- **NIST Secure Hashing Page:** i FIPS di SHA e documenti correlati
- **Whirlpool:** gamma di informazioni su Whirlpool.
- **RSA Laboratories:** estesa collezione di materiale tecnico su RSA a altri argomenti di crittografia.

3.8 Domande di revisione ed esercizi

Domande di revisione

- 3.1 Elenca i tre approcci dell'autenticazione dei messaggi.
- 3.2 Che cos'è il codice di autenticazione del messaggio?
- 3.3 Descrivi brevemente i tre schemi rappresentati nella Figura 3.2.
- 3.4 Quale proprietà deve avere una funzione hash per essere utile nell'autenticazione dei messaggi?
- 3.5 Nel contesto di una funzione hash, che cos'è la funzione di compressione?
- 3.6 Quali sono gli elementi principali di un crittosistema a chiave pubblica?
- 3.7 Elenca e definisci brevemente i tre usi di un crittosistema a chiave pubblica.
- 3.8 Qual è la differenza tra una chiave privata e una chiave segreta?
- 3.9 Che cos'è la firma digitale?

3.10 Che cos'è un certificato a chiave pubblica?

3.11 Come può una cifratura a chiave pubblica essere usata per distribuire una chiave segreta?

Esercizi

3.1 Un algoritmo MAC tra i più utilizzati, conosciuto come algoritmo di autenticazione dei dati, si basa su DES. L'algoritmo è sia una pubblicazione FIPS (FIPS PUB 113) sia uno standard ANSI (X9.17). In pratica, l'algoritmo consiste nell'utilizzare la modalità *cipher block chaining* (CBC) di DES con un vettore di inizializzazione costituito da una serie 0 (Figura 2.9). I dati contigui di 64 bit: P_1, P_2, \dots, P_N . Se necessario, il blocco finale è completato sulla destra con tanti 0 fino a raggiungere la lunghezza di 64 bit. Il MAC corrisponde o all'intero blocco di testo cifrato C_N oppure agli M bit più a sinistra nel blocco, con $16 \leq M \leq 64$. Dimostrare che si perviene allo stesso risultato utilizzando la modalità cipher feedback.

3.2 Considerate una funzione hash a 32 bit definita come la concatenazione di due funzioni di 16 bit: OR esclusivo e OR esclusivo con rotazione, definite come "due semplici funzioni hash" nel Paragrafo 3.2.

- Questo tipo di checksum è in grado di rilevare tutti gli errori causati da un numero dispari di bit errati? Motivate la risposta.
- Questo tipo di checksum è in grado di rilevare tutti gli errori causati da un numero pari di bit errati? Se la risposta è no, caratterizzate le sequenze di bit errati che inducono l'errato funzionamento del checksum.
- Discutete l'efficacia di questa funzione dal punto di vista del suo uso come funzione hash per scopi di autenticazione.

3.3 Ipotizzate che $H(m)$ sia una funzione hash resistente alle collisioni che fa corrispondere un messaggio avente lunghezza in bit arbitraria a un valore hash di n bit. È vero che, per tutti i messaggi x, x' con $x \neq x'$, si ha $H(x) \neq H(x')$? Motivate la risposta.

3.4 a. Considerate la seguente funzione hash. Due messaggi sono nella forma di una sequenza di numeri decimali, $M = (a_1, a_2, \dots, a_n)$. Il valore hash b viene calcolato come

$$\left(\sum_{i=1}^n a_i \right) \bmod n,$$

per qualche valore predefinito di n . Questa funzione hash soddisfa uno qualsiasi dei requisiti per una funzione hash, elencati al Paragrafo 3.2? Argomentate la risposta.

b. Ripetete la parte (a) per la funzione hash

$$b = \left(\sum_{i=1}^n a_i \right) \bmod n.$$

c. Calcolate la funzione hash della parte (b) per $M = 189, 632, 722, 349$ e $n = 989$.

3.5 Questo problema introduce una funzione hash simile idealmente a SHA che opera sulle lettere invece che sui dati binari. È chiamata *toy tetraphash hash* (tth)⁶. Dato un messaggio che consiste

in una sequenza di lettere, tth produce un valore hash che consiste in quattro lettere. Per prima cosa, tth divide il messaggio in blocchi di 16 lettere, ignorando gli spazi, la punteggiatura e l'uso delle maiuscole. Se la lunghezza del messaggio non è divisibile per 16, viene riempito con dei valori nulli. Viene mantenuto un totale di esecuzione di quattro numeri, che parte con il valore $(0, 0, 0, 0)$; questo è l'ingresso della funzione di compressione per elaborare il primo blocco. La funzione di compressione consiste di due iterazioni. **Iterazione 1:** prendete il prossimo blocco di testo e diponetelo come un blocco di testo 4×4 ordinato per righe (*row-wise*) e convertitelo in numeri ($A = 0, B = 1, \dots$). Ad esempio, per il blocco ABCDEFGHIJKLMNOP; si ha:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Aggiungete poi a ciascuna colonna mod 26 e aggiungete il risultato della somma precedente al totale di esecuzione, mod 26. In questo esempio, il totale di esecuzione è $(24, 2, 6, 10)$. **Iterazione 2:** usando la matrice dell'Iterazione 1, spostate la prima riga a sinistra di una posizione, la seconda riga a sinistra di due posizioni, la terza riga di tre posizioni e invertite l'ordine per la quarta riga. Nel nostro esempio:

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Ora, si aggiunga a ciascuna colonna mod 26 e si aggiunga il risultato della precedente somma al totale di esecuzione. Il nuovo totale di esecuzione è $(5, 7, 9, 11)$. Il totale di esecuzione è ora l'ingresso della prima iterazione della funzione di compressione del successivo blocco di testo. Dopo che il blocco finale è stato elaborato, si converta il totale di esecuzione finale in lettere. Ad esempio, se il messaggio è ABCDEFGHIJKLMNOP, allora hash è FHJL.

- Disegnate figure simili alle Figura 3.4 e 3.5, per rappresentare la logica completa di tth e la logica della funzione di compressione.
- Calcolate la funzione hash per il messaggio di 48 lettere: "I leave twenty million dollars to my friend cousin Bill".
- Per dimostrare la debolezza di tth, trovate un blocco che produca lo stesso hash di quello appena calcolato. *Suggerimento:* usate molte A.
- Una funzione hash può essere utilizzata per costruire un cifrario a blocchi con una struttura simile a DES. Come è possibile, visto che la funzione hash è unidirezionale mentre un cifrario a blocchi deve essere reversibile (per consentire la decifratura)?
- Prima della scoperta di un qualunque specifico schema di cifratura a chiave pubblica, come ad esempio RSA, si formulò una dimostrazione che provava come la cifratura a chiave pubblica fosse teoricamente possibile. Considerate le funzioni $f_1(x_1) = z_1$; $f_2(x_2, y_2) = z_2$; $f_3(x_3, y_3) = z_3$, dove tutti i valori sono interi, con $1 \leq x_i, y_i, z_i \leq N$. La funzione f_1 può essere rappresentata mediante un vettore M_1 di lunghezza N , in cui il k -esimo elemento è il valore di $f_1(k)$. In maniera analoga, f_2 e f_3 possono essere rappresentate mediante matrici $N \times N$, denominate M_2 and M_3 . Lo scopo è quello di rappresentare i processi di cifratura/decifratura mediante tabelle di consulta-

⁶ Si ringrazia William K. Mason, dello staff del giornale *The Cryptogram*, per aver fornito l'esempio.

zione (*table look-up*) per tabelle con valori di N molto grandi che, anche se enormi e difficili da gestire, potrebbero essere, in linea di principio, costruite. Lo schema opera in questo modo: costruire M_1 con una permutazione casuale di tutti gli interi compresi fra 1 e N ; in altre parole, ogni intero appare una sola volta all'interno di M_1 . Costruire M_2 in modo che ciascuna riga contenga una permutazione casuale dei primi N interi. Infine, riempire M_3 verificando la seguente condizione:

$$f_3(f_2(f_1(k), p), k) = p \quad \text{per ogni } k, p \text{ con } 1 \leq k, p \leq N$$

In altre parole:

1. M_1 riceve in ingresso k e produce come risultato x ;
2. M_2 riceve in ingresso x e p e produce come risultato z ;
3. M_3 riceve in ingresso z e k e produce come risultato p .

Una volta costruite, queste tre tabelle vengono rese pubbliche.

- a. Deve essere chiaro che è possibile costruire M_3 in modo da soddisfare la condizione precedente. A titolo di esempio, inserite valori in M_3 per il seguente semplice caso:

$M_1 =$	$M_2 =$	$M_3 =$
5 4 2 3 1	5 2 3 4 1 4 2 5 1 3 1 3 2 4 5 3 1 4 2 5 2 5 3 4 1	

Convenzione: l' i -esimo elemento di M_1 corrisponde a $k = i$. La i -esima riga di M_2 corrisponde a $x = i$; la j -esima colonna di M_2 corrisponde a $p = j$. La i -esima riga di M_3 corrisponde a $z = i$; la j -esima colonna di M_3 corrisponde a $k = j$.

- b. Descrivete l'uso di queste tabelle per eseguire cifratura e decifratura fra due utenti.
 c. Dimostrate come questo costituisca uno schema sicuro.
- 3.8 Eseguite i processi di cifratura e decifratura utilizzando l'algoritmo RSA, come nella Figura 3.9, con i seguenti valori:

- a. $p = 3; q = 11; e = 7; M = 5$
- b. $p = 5; q = 11; e = 3; M = 9$
- c. $p = 7; q = 11; e = 17; M = 8$
- d. $p = 11; q = 13; e = 11; M = 7$
- e. $p = 17; q = 31; e = 7; M = 2$

Suggerimento: la decifratura è meno complessa di quanto possa sembrare; utilizzate qualche stratagemma.

- 3.9 In un sistema a chiave pubblica basato su RSA, viene intercettato il testo cifrato $C = 10$ inviato a un utente la cui chiave pubblica è $e = 5, n = 35$. Qual è il corrispondente testo in chiaro M ?
- 3.10 In un sistema basato su RSA, la chiave pubblica di un dato utente è $e = 31, n = 3599$. Qual è la chiave privata di questo utente?
- 3.11 Supponiamo di avere un insieme di blocchi cifrati con l'algoritmo RSA e di non possedere la chiave privata. Ipotizzate che $n = pq$, e che e sia la chiave pubblica. Supponete anche che qualcuno ci porti a conoscenza del fatto che uno dei blocchi di testo in chiaro ha un fattore comune con n . Questo può essere di qualche aiuto?

- 3.12 Mostrate come sia possibile rappresentare RSA mediante le matrici M_1, M_2 , e M_3 dell'Esercizio 3.4.

- 3.13 Considerate il seguente schema:

- a. Scegliere un numero dispari E .
- b. Scegliere due numeri primi, P e Q , dove $(P - 1)(Q - 1) - 1$ è divisibile in parti uguali per E .
- c. Moltiplicare P e Q per ottenere N .
- d. Calcolare:

$$D = \frac{(P - 1)(Q - 1)(E - 1) + 1}{E}$$

Questo schema è equivalente a RSA? Motivate sia la risposta affermativa che quella negativa.

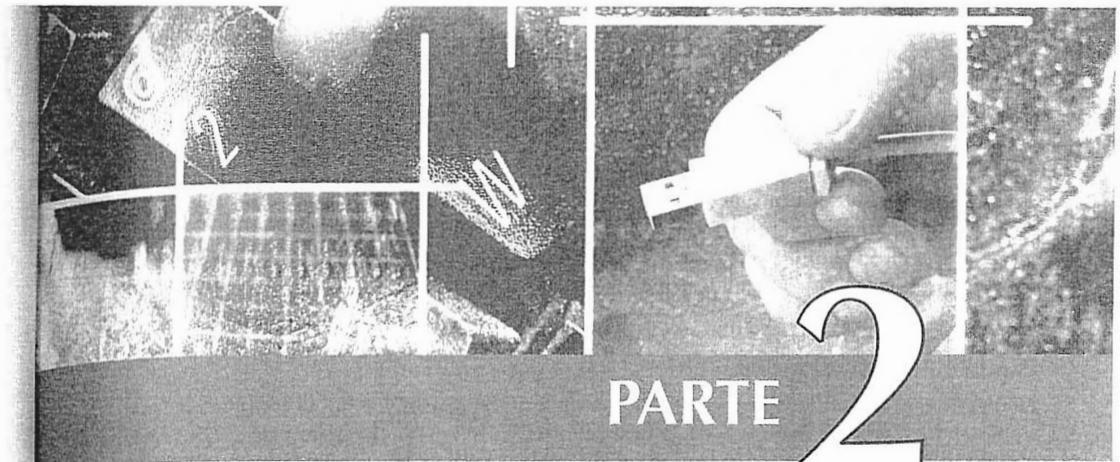
- 3.14 Ipotizzate di utilizzare RSA con una chiave nota per costruire una funzione hash unidirezionale. Successivamente, elaborate un messaggio formato da una sequenza di blocchi nel seguente modo: cifrare il primo blocco, eseguire l'OR esclusivo del risultato ottenuto con il secondo blocco e cifrare nuovamente, e così via. Mostrate che questo schema non è sicuro risolvendo il seguente problema. Sia dato un messaggio composto da due blocchi B_1 e B_2 , e la sua corrispondente codifica hash

$$\text{RSAH}(B_1, B_2) = \text{RSA}(\text{RSA}(B_1) \oplus B_2)$$

Dato un blocco C_1 , scegliete C_2 così che $\text{RSAH}(C_1, C_2) = \text{RSAH}(B_1, B_2)$. La funzione hash quindi non soddisfa la resistenza debole alla collisione.

- 3.15 Supponete che Bob usi il crittosistema RSA con un modulo n molto grande per cui la fattorizzazione non si può trovare in una ragionevole quantità di tempo. Ipotizzate che Alice invii un messaggio a Bob rappresentando ciascun carattere alfabetico come un intero da 0 a 25 ($A \rightarrow 0, \dots, Z \rightarrow 25$) e poi cifrando ciascun numero separatamente usando RSA con un grande e e un grande n . Questo metodo è sicuro? Se non lo è, descrivete l'attacco più efficiente nei confronti di questo metodo di cifratura.

- 3.16 Considerate uno schema Diffie-Hellman con numero primo comune $q = 11$ e radice primitiva $\alpha = 2$.
- a. Se un utente A possiede chiave pubblica $Y_A = 9$, qual è la chiave privata X_A di A?
 - b. Se un utente B possiede chiave pubblica $Y_B = 3$, qual è la chiave segreta condivisa K ?



PARTE 2

Applicazioni di sicurezza di rete

Nella Prima Parte sono stati esaminati diversi cifrari e il loro uso per garantire la riservatezza, l'autenticazione, le scambi delle chiavi e le funzioni correlate. La seconda parte è dedicata a una panoramica dei più importanti strumenti e delle principali applicazioni per la sicurezza di rete, che fanno uso di queste funzioni. Queste applicazioni possono essere usate nell'ambito di una singola rete, di una intranet aziendale o di Internet.

Capitolo 4 Applicazioni di autenticazione

Il Capitolo 4 presenta due tra le più importanti specifiche di autenticazione attualmente utilizzate: Kerberos e X.509. Kerberos è un protocollo di autenticazione basato su cifratura convenzionale che ha ottenuto un ampio supporto ed è utilizzato in molteplici sistemi. X.509 specifica un algoritmo di autenticazione e definisce funzionalità per l'utilizzo e la gestione di certificati. X.509 consente agli utenti di ottenere certificati a chiave pubblica in modo che una comunità di utenti possa avere fiducia nella validità delle corrispondenti chiavi. Questa funzionalità è impiegata come elemento per realizzare numerose applicazioni.

Capitolo 5 Sicurezza della posta elettronica

La posta elettronica è l'applicazione distribuita maggiormente utilizzata; è naturale quindi che vi sia un crescente interesse nell'arricchirla con servizi di autenticazione e riservatezza. Il Capitolo 5 illustra i due approcci probabilmente destinati a dominare l'ambito della sicurezza della posta elettronica nel prossimo futuro: PGP e S/MIME. PGP (*pretty good privacy*) è uno schema ampiamente utilizzato che non dipende da alcuna organizzazione o autorità. Per questo è adatto a un uso personale e individuale, oltre che essere incorporato in configurazioni di rete gestite da organizzazioni. S/MIME (*secure/multipurpose internet mail extension*) è stato specificatamente progettato per divenire uno standard Internet.

Capitolo 6 Sicurezza IP

Sia in Internet che nelle intranet private, il protocollo Internet (IP) ricopre un ruolo centrale. Di conseguenza, la sicurezza a livello IP costituisce un fattore importante nella protezione degli schemi di sicurezza di rete. Il Capitolo 6 analizza gli schemi di sicurezza IP sviluppati sia per l'attuale IP sia per l'emergente IP di nuova generazione, conosciuto come IPv6.

Capitolo 7 Sicurezza web

La rapidissima crescita nell'utilizzo del World Wide Web per applicazioni di commercio elettronico e per la distribuzione delle informazioni ha fatto nascere l'esigenza di affrontare il problema della sicurezza web. Il Capitolo 7 offre una panoramica di questa importante area della sicurezza ed analizza due standard rilevanti: SSL (*secure sockets layer*) e SET (*secure electronic transaction*).

Capitolo 8 Sicurezza della gestione di rete

Il crescente utilizzo di sistemi per la gestione di rete per il controllo di reti fornite da differenti produttori, ha generato l'esigenza di corrispondenti funzionalità di sicurezza. Il Capitolo 8 si focalizza sullo schema più utilizzato per la gestione di reti di differenti produttori: SNMP (*simple network management protocol*). La versione 1 di SNMP comprende solo una rudimentale funzionalità di autenticazione basata su password. SNMPv2 fornisce funzionalità più avanzate, mentre SNMPv3 fornisce un insieme completo di funzionalità di sicurezza per garantire riservatezza e autenticazione, utilizzabili in combinazione con SNMPv1 o SNMPv2.

Capitolo 4

Applicazioni di autenticazione

In questo capitolo verranno illustrate alcune delle funzioni sviluppate per fornire un supporto all'autenticazione a livello di applicazione e alle firme digitali.

Il capitolo inizia illustrando Kerberos, uno dei primi e più utilizzati servizi di autenticazione; successivamente verrà presentato il servizio di autenticazione X.509. Tale standard è rilevante sia come componente del servizio di directory che fornisce sia perché viene utilizzato come elemento costitutivo in altri standard, come S/MIME, trattato nel Capitolo 5.

Infine sarà esaminato il concetto di infrastruttura per la chiave pubblica.

4.1 Kerberos

Kerberos¹ è un servizio di autenticazione sviluppato al MIT nell'ambito del progetto Athena. Il problema affrontato da Kerberos è il seguente: si consideri un ambiente aperto e distribuito in cui gli utenti, collegati a workstation, desiderano accedere a servizi forniti da server distribuiti sulla rete. I server devono essere in grado di consentire gli accessi solo a utenti autorizzati e di autenticare le richieste di servizi. In questo ambiente non è possibile fare affidamento sulle workstation per identificare correttamente gli utenti per l'utilizzo dei servizi di rete. In particolare, esistono tre principali tipologie di minacce.

- Un intruso può ottenere l'accesso a una data workstation e fingere di essere un altro utente che opera da quella workstation.
- Un utente può modificare l'indirizzo di rete di una workstation e inoltrare da questa richieste in modo che sembrino inviate dalla workstation cui corrisponde l'indirizzo modificato.

¹ "Secondo la mitologia greca, cane con molte teste, generalmente tre, forse con una coda di serpente, guardiano dell'entrata dell'Ade." Dal *Dictionary of Subjects and Symbols in Art*, di James Hall, Harper & Row, 1979. In analogia con il Kerberos (in italiano Cerbero) della mitologia greca, dotato di tre teste, il moderno Kerberos è stato concepito per essere costituito da tre componenti per proteggere una porta di rete: autenticazione, accounting e audit. Le ultime due componenti non sono mai state implementate.

- Un avversario può ascoltare gli scambi che avvengono sulla rete e utilizzare un attacco di replay per entrare in un server o per effettuare azioni distruttive.

In ognuno dei casi sopra illustrati, esiste la possibilità che un utente non autorizzato sia in grado di ottenere accessi a servizi e dati per cui non possiede la necessaria autorizzazione. Per ovviare a tali inconvenienti, invece di definire protocolli di autenticazione complessi per ciascun server, Kerberos fornisce il server di autenticazione centralizzato la cui funzione è quella di autenticare gli utenti ai server e viceversa. Kerberos, a differenza della maggior parte degli altri schemi di autenticazione descritti in questo testo, si basa unicamente sulla crittografia convenzionale, e non su quella a chiave pubblica.

Due sono le versioni di Kerberos comunemente utilizzate: la versione 4 [MILL88, STEI88] e la versione 5 [KOHL94], che ovvia ad alcuni limiti di sicurezza della versione 4 ed è stata pubblicata come Proposed Internet standard (RFC 1510)².

Motivazioni

Se un insieme di utenti dispone di personal computer dedicati, senza connessioni a una rete, è possibile proteggere le risorse e i file degli utenti proteggendo fisicamente ciascun calcolatore. Nel caso in cui gli utenti operino tramite un sistema centralizzato a condivisione di tempo (*time-sharing*), sarà compito del sistema operativo garantire la sicurezza del sistema. Il sistema operativo può applicare politiche di controllo dell'accesso basate sull'identità dell'utente e utilizzare la procedura di connessione (*logon*) per identificare gli utenti.

Oggi, nessuno di questi approcci è comunemente utilizzato. L'approccio più diffuso prevede invece un'architettura distribuita formata da workstation utente dedicate (client) e da server distribuiti o centralizzati. In tale scenario, si possono prefigurare tre differenti approcci per garantire la sicurezza.

1. Fare affidamento su ciascuna workstation client per garantire l'identità dei propri utenti e fare affidamento sui server per realizzare una politica di sicurezza basata sull'identità degli utenti (ID).
2. Richiedere che i sistemi client si autentichino ai server, ma fare affidamento sui client per quanto riguarda l'identità dei loro utenti.
3. Richiedere che l'utente fornisca prova della sua identità tutte le volte che richiede un servizio. Richiedere inoltre che i server forniscano prova della loro identità ai client.

In un ambiente chiuso e di piccole dimensioni, in cui tutti i sistemi sono di proprietà ed amministrati da un'unica organizzazione, può essere sufficiente la prima o forse la seconda strategia³. Al contrario, in un ambiente più aperto, in cui è possibile connettersi in rete con altre macchine, è necessario il terzo approccio per proteggere le informazioni utente e le risorse che risiedono sul server. Questo tipo di approccio è quello fornito da Kerberos. Kerberos si basa su un'architettura client/server distribuita e utilizza uno o più server Kerberos per fornire il servizio di autenticazione.

² Le versioni dalla 1 alla 3 erano a uso interno. La 4 è la versione "originale" di Kerberos.

³ Anche in un ambiente chiuso esiste comunque la minaccia di un attacco da parte di un impiegato scontento.

Il primo rapporto pubblicato su Kerberos [STEI88] conteneva la seguente lista di requisiti.

- **Sicurezza.** Un intruso, in ascolto sulla rete, non dovrebbe essere in grado di ottenere le informazioni necessarie a impersonare un utente legittimo. Più in generale, Kerberos dovrebbe essere sufficientemente robusto da non costituire il punto debole per un avversario.
- **Affidabilità.** L'indisponibilità di Kerberos implica la non disponibilità di tutti i servizi che lo utilizzano per il controllo dell'accesso. Per questo Kerberos dovrebbe essere estremamente affidabile e utilizzare un'architettura con più server distribuiti, in cui un sistema possa agire da sistema di back-up per un altro.
- **Trasparenza.** Idealmente, il processo di autenticazione dovrebbe essere trasparente all'utente, ad esclusione della richiesta di inserimento della password.
- **Scalabilità.** Il sistema dovrebbe essere in grado di gestire un numero elevato di client e server. Questo requisito suggerisce l'utilizzo di un'architettura modulare e distribuita.

Per soddisfare questi requisiti, lo schema globale di Kerberos prevede una terza parte fidata che fornisce il servizio di autenticazione, e utilizza un protocollo basato su quello proposto da Needham e Schroeder [NEED78]. Tale parte è fidata nel senso che client e server hanno fiducia in Kerberos per quanto concerne la mediazione della loro reciproca autenticazione. Ipotizzando che il protocollo di Kerberos sia ben progettato, il servizio di autenticazione è sicuro nella misura in cui è sicuro il server Kerberos⁴.

Versione 4 di Kerberos

Per fornire il servizio di autenticazione, la versione 4 di Kerberos utilizza DES all'interno di un protocollo piuttosto articolato. Considerando il protocollo nel suo insieme, è difficile capire la necessità dei molti elementi di cui è costituito. Per questa ragione, nel prosieguo della trattazione verrà adottata la stessa strategia utilizzata da Bill Bryant del progetto Athena [BRYA88] che consiste nel giungere al protocollo completo partendo da alcuni dialoghi ipotetici. A ciascun dialogo la complessità aumenta per ovviare alle vulnerabilità, dal punto di vista della sicurezza, emerse nel dialogo precedente.

Dopo aver analizzato il protocollo, si illustreranno alcuni altri aspetti relativi alla versione 4.

Un dialogo di autenticazione semplice

In un ambiente di rete non protetto, un qualsiasi client può richiedere un servizio a un qualsiasi server. In questo caso il rischio più evidente dal punto di vista della sicurezza è quello

⁴ Si ricordi che non si può dare per scontata la sicurezza del server Kerberos, ma questo deve essere accuratamente protetto (per esempio, in una stanza chiusa a chiave). È bene a tal proposito ricordare il destino di Kerberos nella mitologia greca. Euristeo ordinò a Ercole di catturarlo, come sua dodicesima fatica: "Ercole trovò il grosso cane legato alla sua catena e lo afferrò per la gola. Immediatamente le tre teste tentarono un attacco e Kerberos sferrò tutta la zona circostante con la sua potente coda. Ercole si attaccò alla coda con tutte le sue forze e Kerberos cadde in uno stato di incoscienza. Euristeo rimase sorpreso di vedere Ercole vivo; quando vide le tre teste bavose ed il grosso cane cui appartenevano si spaventò a morte e si rinchiuse al sicuro nella sua giara di bronzo". Da *The Hamlyn Concise Dictionary of Greek and Roman Mythology*, di Michael Stapleton, Hamlyn, 1982.

dell'impersonificazione: un avversario può fingere di essere un altro client e ottenere così privilegi non autorizzati sulle macchine server. Per contrastare questa minaccia, i server devono essere in grado di convalidare le identità dei client che richiedono un servizio. A ciascun server può essere richiesto di effettuare tale operazione per ogni interazione tra client e server ma, in un ambiente aperto, questo appesantirebbe molto le attività del server.

Un'altra alternativa consiste nell'utilizzare un server di autenticazione (AS, *authentication server*) che sia a conoscenza delle password di tutti gli utenti e le memorizzi in una base di dati centralizzata. Inoltre, l'AS condivide con ciascun server una chiave segreta unica. Queste chiavi sono distribuite fisicamente o tramite qualche modalità sicura. Si consideri il seguente dialogo ipotetico⁵:

- (1) C → AS: $ID_C \| P_C \| ID_V$
 - (2) AS → C: *Ticket*
 - (3) C → V: $ID_C \| \text{Ticket}$
- Ticket* = $E(K_v, [ID_C \| AD_C \| ID_V])$

dove

C = client

AS = server di autenticazione

V = server

ID_C = identificatore dell'utente connesso a C

ID_V = identificatore di V

P_C = password dell'utente connesso a C

AD_C = indirizzo di rete di C

K_v = chiave di cifratura segreta condivisa da AS e V

$\|$ = operatore di concatenazione

In questo scenario, l'utente si connette a una workstation e richiede l'accesso a un server V. Il modulo client C nella workstation cui l'utente è connesso richiede all'utente la password e invia all'AS un messaggio contenente l'ID dell'utente, l'ID del server, e la password dell'utente. L'AS consulta la propria base di dati per verificare se l'utente ha fornito la password corrispondente al proprio ID e se l'utente ha il permesso di accedere al server V. Se entrambi i controlli sono superati, l'AS riconosce l'utente come autentico e deve a questo punto convincere il server della stessa cosa. A tal fine, l'AS genera un ticket contenente l'ID e l'indirizzo di rete dell'utente e l'ID del server. Il ticket è cifrato mediante la chiave segreta condivisa dall'AS e dal server in questione, e viene quindi spedito a C. Il ticket non può essere modificato né da C né da un avversario, in quanto è trasmesso in forma cifrata.

Una volta ricevuto il ticket, C può richiedere un servizio a V. C invia un messaggio a V contenente il proprio ID e il ticket. V decifra il ticket e verifica che l'ID utente contenuto

nel ticket coincida con l'ID utente non cifrato contenuto nel messaggio. Se i due ID sono identici, il server considera l'utente come autenticato e gli fornisce il servizio richiesto.

Ciascuna delle componenti del messaggio (3) è rilevante. Il ticket è cifrato per prevenire una sua modifica o contraffazione. L'ID del server (ID_V) è inserito nel ticket affinché il server possa verificare di aver decifrato il ticket in modo corretto. L'ID del client (ID_C) è inserito nel ticket per indicare che il ticket è stato generato per conto di C. Infine, AD_C serve per contrastare la seguente minaccia. Un avversario potrebbe intercettare il ticket trasmesso nel messaggio (2) e utilizzare ID_C per trasmettere un messaggio con la struttura di (3) da un'altra workstation. In questo caso il server riceverebbe un ticket valido il cui ID contenuto coincide con l'ID utente e concederebbe quindi l'accesso all'utente sull'altra workstation. Per prevenire questo tipo di attacco, l'AS inserisce nel ticket l'indirizzo di rete da cui è stata inviata la richiesta originaria. In questo modo il ticket è valido solo se trasmesso dalla stessa workstation che ha inizialmente richiesto il ticket stesso.

Un dialogo di autenticazione più sicuro

Lo scenario descritto nel precedente paragrafo risolve alcuni problemi connessi all'autenticazione in un ambiente di rete aperto, ma non li elimina completamente. I problemi più rilevanti sono essenzialmente due. In primo luogo, si vorrebbe minimizzare il numero di richieste di password all'utente. Si assume che ogni ticket possa essere utilizzato una sola volta. Se alla mattina l'utente si connette a una workstation per controllare la sua posta elettronica residente su un dato server di posta, dovrà fornire una password per ottenere un ticket per il server di posta. Se l'utente vuole poi controllare la posta più volte durante la giornata, dovrà ogni volta re-immettere la password. La situazione può essere migliorata rendendo il ticket riutilizzabile. Per ogni singola sessione di connessione, la workstation può memorizzare il ticket per il server di posta dopo la sua ricezione e riutilizzarlo per conto dell'utente ogni qual volta questi abbia necessità di connettersi al server di posta.

Comunque, anche con questo schema l'utente ha bisogno di un nuovo ticket ogni volta che richiede un servizio diverso. Ad esempio, la prima volta che l'utente accede a un server di stampa, a un server di posta o a un file server deve sempre inserire la propria password in quanto è necessario uno specifico ticket per ciascuna tipologia di accesso richiesta.

Il secondo problema è che lo scenario descritto in precedenza prevede la trasmissione in chiaro della password (messaggio 1). Tramite l'ascolto della rete si potrebbe intercettare la password e utilizzare uno qualsiasi dei servizi che possono essere richiesti dall'utente cui la password corrisponde.

Per risolvere questi ulteriori problemi, vengono introdotti uno schema, che evita la trasmissione in chiaro delle password, e un nuovo server, chiamato distributore di ticket (TGS, *ticket-granting server*). Il nuovo, ma ancora ipotetico, scenario è il seguente.

Una sola volta per sessione di connessione:

- (1) C → AS: $ID_C \| ID_{tg}$
- (2) AS → C: $E(K_v, \text{Ticket}_{tg})$

Una sola volta per tipologia di servizio:

- (3) C → TGS: $ID_C \| ID_V \| \text{Ticket}_{tg}$
- (4) TGS → C: Ticket_v

⁵ La parte a sinistra dei due punti indica il mittente e il ricevente; la parte a destra indica il contenuto del messaggio e il simbolo || indica la concatenazione.

Una sola volta per sessione di servizio:

(5) $C \rightarrow V: ID_C \| Ticket_v$

$$Ticket_{TGS} = E(K_{TGS}[ID_C \| AD_C \| ID_{TGS} \| TS_1 \| \text{Tempo di validità}_1]$$

$$Ticket_v = E(K_v[ID_C \| AD_C \| ID_v \| TS_2 \| \text{Tempo di validità}_2]$$

Il nuovo servizio, TGS, emette un ticket per gli utenti che sono stati autenticati dall'AS. In primo luogo, l'utente richiede all'AS un ticket di tipo ticket-granting ($Ticket_{TGS}$). Questo ticket viene memorizzato dal modulo client C della workstation cui l'utente è connesso. Ogni qualvolta l'utente richiede di accedere a un nuovo servizio, C si rivolge al TGS, utilizzando il ticket per autenticarsi. Il TGS emette quindi un ticket per il servizio in questione. Il client salva ciascun ticket di tipo service-granting ricevuto e lo utilizza per autenticare il proprio utente ai server ogni volta che viene richiesto un certo servizio. Consideriamo ora in dettaglio questo schema.

1. Il client richiede un ticket di tipo ticket-granting per conto dell'utente inviando all'AS l'ID dell'utente congiuntamente all'ID del TGS, per indicare che si tratta di una richiesta di utilizzo dei servizi messi a disposizione dal TGS.
2. L'AS risponde con un ticket cifrato con una chiave (K_C) derivata dalla password dell'utente. Quando la risposta perviene al client C, quest'ultimo chiede all'utente l'invio della password, genera la chiave, e tenta di decifrare il messaggio in arrivo. Se l'utente ha fornito la password corretta, il ticket è decifrato con successo.

Solo l'utente legittimo può decifrare con successo il ticket, dal momento che è l'unico a conoscenza della password corretta. In questo schema, la password viene utilizzata per ottenere da Kerberos le credenziali senza la necessità di trasmettere la password in chiaro. Il ticket contiene l'ID e l'indirizzo di rete dell'utente e l'ID del TGS. Questo in analogia con quanto avveniva nel primo scenario. L'idea è che questo ticket possa essere utilizzato dal client per richiedere molteplici ticket di tipo service-granting. Questi devono quindi essere riutilizzabili. Allo stesso tempo, ci si vuole cautelare dal pericolo che un avversario sia in grado di intercettare il ticket e utilizzarlo. Si consideri il seguente scenario. L'avversario intercetta il ticket ed attende fino a quanto l'utente si scollega dalla propria workstation. A questo punto l'avversario o ottiene l'accesso alla workstation dell'utente o configura la propria con lo stesso indirizzo di rete dell'utente cui appartiene il ticket. L'avversario sarà in grado di riutilizzare il ticket per frodare il TGS. Per far fronte a questo pericolo, il ticket comprende un istante temporale, denominato timestamp, che indica la data e l'istante di tempo in cui ha avuto luogo la generazione del ticket, e un tempo di validità del ticket, che indica il lasso di tempo durante il quale il ticket è valido (per esempio, otto ore). In questo modo, il client possiede un ticket riutilizzabile e non ha bisogno di richiedere all'utente la password ogni volta che questi richiede un nuovo servizio. Infine, si noti che il ticket-granting è cifrato mediante una chiave segreta conosciuta solo dall'AS e dal TGS. Tale cifratura previene alterazioni del ticket. Il ticket è cifrato una seconda volta con una chiave basata sulla password dell'utente. Questo assicura che il ticket possa essere decifrato solo dall'utente appropriato, fornendo quindi un meccanismo di autenticazione.

Quando il client possiede un ticket-granting, l'accesso a un qualsiasi server può essere ottenuto mediante i Passi 3 e 4.

3. Il client C richiede un ticket di tipo service-granting per conto dell'utente. A tal scopo, C invia un messaggio al TGS contenente l'ID dell'utente, l'ID del servizio che si vuole richiedere e il ticket di tipo ticket-granting.

4. Il TGS decifra il ticket ricevuto e verifica se la decifratura ha avuto successo dal fatto che sia presente il proprio ID. Il TGS verifica inoltre che il ticket non sia scaduto. A questo punto, per autenticare l'utente, confronta l'ID e l'indirizzo di rete dell'utente con le informazioni contenute nel ticket. Se l'utente ha il diritto di accedere a V, il TGS emette un ticket per concedere l'accesso al servizio richiesto.

Il ticket di tipo service-granting ha la stessa struttura di quello di tipo ticket-granting. Tale uniformità di struttura è conseguenza del fatto che, essendo il TGS un server, per autenticare un client al TGS serviranno gli stessi elementi richiesti per autenticare un client a un server qualunque. Anche in questo caso, il ticket contiene un timestamp e un tempo di validità. Se l'utente ha necessità di accedere allo stesso servizio in tempi diversi, il client può semplicemente utilizzare il ticket di tipo service-granting precedentemente acquisito senza la necessità di dover richiedere all'utente la password. Al fine di prevenire alterazioni, il ticket è cifrato con la chiave segreta (K_v) conosciuta solo dal TGS e dal server.

Infine, una volta ottenuto un certo ticket di tipo service-granting, il client può ottenere l'accesso al corrispondente servizio tramite il Passo 5.

5. Il client C richiede l'accesso a un servizio per conto dell'utente. A tal fine, C trasmette al server un messaggio contenente l'ID dell'utente e il ticket di tipo service-granting. Il server procede all'autenticazione sulla base del contenuto del ticket.

Questo nuovo scenario soddisfa i due requisiti legati al richiedere una sola volta la password per ogni sessione utente, assicurando nel contempo la protezione delle password.

Dialogo di autenticazione della versione 4

Lo scenario appena illustrato migliora la sicurezza rispetto al primo caso, ma lascia ancora aperti due problemi. Il punto cruciale del primo problema è connesso al tempo di validità del ticket di tipo ticket-granting. Se il tempo di validità è molto limitato (ad esempio, nell'ordine dei minuti), allora all'utente sarà richiesta molte volte la password. Viceversa, se il tempo di validità è elevato (ad esempio, nell'ordine delle ore), l'avversario potrebbe ascoltare la rete e intercettare una copia del ticket di tipo ticket-granting aspettando poi che l'utente legittimo si scolleghi. A questo punto l'avversario potrebbe falsificare l'indirizzo di rete dell'utente legittimo e inviare al TGS il messaggio descritto al passo (3). Questa azione darebbe all'avversario un accesso illimitato alle risorse ed ai file cui il legittimo utente ha accesso.

Analogamente, se l'avversario intercetta un ticket di tipo service-granting e lo utilizza prima della sua scadenza, può ottenere l'accesso al servizio corrispondente.

Per tali ragioni, si richiede un ulteriore requisito. Un servizio di rete (il TGS o un servizio applicativo) deve essere in grado di dimostrare che la persona che utilizza il ticket è la stessa per cui il ticket è stato emesso.

Il secondo problema deriva dalla necessità che i server si autentichino agli utenti, altrimenti un avversario potrebbe sabotare la configurazione in modo che i messaggi inviati a un server siano mandati da un'altra parte. Il server illegittimo sarebbe quindi in grado di

agire come se fosse il server reale e potrebbe intercettare una qualsiasi informazione proveniente dall'utente e impedire all'utente di utilizzare il servizio legittimo.

Esamineremo in seguito questi problemi riferendoci alla Tabella 4.1, che mostra l'effettivo protocollo di Kerberos.

- (1) $C \rightarrow AS \quad ID_c \| ID_{gs} \| TS_1$
- (2) $AS \rightarrow C \quad E(K_c, [K_{c,gs}, ID_{gs} \| TS_2 \| \text{Tempo di validità}_2 \| \text{Ticket}_{tg}])$
 $\text{Ticket}_{tg} = E(K_{tg}, [K_{c,gs} \| ID_c \| AD_c \| ID_{gs} \| TS_2 \| \text{Tempo di validità}_2])$

(a) Scambio relativo al servizio di autenticazione: per ottenere un ticket di tipo ticket-granting

- (3) $C \rightarrow TGS \quad ID_v \| \text{Ticket}_{tg} \| \text{Autenticatore}_c$
- (4) $TGS \rightarrow C \quad E(K_{c,gs}, [K_{c,v}, ID_c \| TS_1 \| \text{Ticket}_v])$
 $\text{Ticket}_v = E(K_v, [K_{c,gs} \| ID_c \| AD_c \| ID_v \| TS_1 \| \text{Tempo di validità}_1])$
 $\text{Ticket}_v = E(K_v, [K_{c,v}, ID_c \| AD_c \| ID_v \| TS_1 \| \text{Tempo di validità}_1])$
 $\text{Autenticatore}_c = E(K_{c,gs}, [ID_c \| AD_c \| TS_1])$

(b) Scambio relativo al servizio di distribuzione di ticket: per ottenere un ticket di tipo service-granting

- (5) $C \rightarrow V \quad \text{Ticket}_v \| \text{Autenticatore}_c$
- (6) $V \rightarrow C \quad E(K_{c,v}, [TS_5 + 1])$ (per autenticazione reciproca)
 $\text{Ticket}_v = E(K_v, [K_{c,v}, ID_c \| AD_c \| ID_v \| TS_1 \| \text{Tempo di validità}_1])$
 $\text{Autenticatore}_c = E(K_{c,v}, [ID_c \| AD_c \| TS_1])$

(c) Scambio relativo all'autenticazione tra client e server: per ottenere un servizio

Tabella 4.1 Sintesi dello scambio di messaggi nella versione 4 di Kerberos.

Si consideri in primo luogo il problema dei ticket di tipo ticket-granting intercettati e la necessità di determinare che il client che presenta un ticket sia lo stesso per cui il ticket era stato generato. La minaccia è costituita dal fatto che un avversario può rubare il ticket e utilizzarlo prima della sua scadenza. Tale problema può essere evitato facendo in modo che l'AS fornisca al client ed al TGS una qualche informazione riservata mediante una modalità sicura. In tal modo il client può provare al TGS la propria identità rivelandogli nuovamente, mediante una modalità sicura, l'informazione segreta ricevuta dall'AS. Un modo efficiente per realizzare questo schema è utilizzare come informazione riservata una chiave di cifratura chiamata, in Kerberos, chiave di sessione.

La Tabella 4.1a illustra la tecnica per la distribuzione delle chiavi di sessione. Come nello scenario precedente, il client invia un messaggio all'AS per richiedere l'accesso al TGS. L'AS risponde con un messaggio contenente il ticket. Tale messaggio è cifrato con una chiave (K_c) ottenuta dalla password dell'utente. Il messaggio cifrato contiene inoltre una copia della chiave di sessione, $K_{c,gs}$, dove il pedice indica che si tratta di una chiave di sessione per C e per il TGS. Dal momento che la chiave di sessione è contenuta all'interno del

messaggio cifrato con K_c solo il client corrispondente all'utente in questione sarà in grado di leggerla. La stessa chiave di sessione è contenuta anche all'interno del ticket, che può essere letto solo dal TGS. La chiave di sessione è quindi inviata in modalità sicura sia a C sia al TGS.

Prima di procedere con la trattazione, è opportuno notare come alla prima fase del dialogo siano state aggiunte alcune informazioni. Il messaggio (1) contiene un timestamp, in modo che l'AS possa verificare che il messaggio è appropriato rispetto al tempo di ricezione. Il messaggio (2) contiene diversi elementi del ticket in una forma accessibile a C. Questo consente a C di confermare che il ticket è per il TGS e di conoscerne la scadenza.

Una volta ottenuti il ticket e la chiave di sessione, C è pronto a interagire con il TGS. Come nel caso precedente, C invia al TGS un messaggio contenente il ticket e l'ID del servizio richiesto (messaggio 3 nella Tabella 4.1b). Inoltre, C invia un autenticatore, che contiene l'ID e l'indirizzo dell'utente di C e un timestamp. A differenza del ticket, che è riutilizzabile, l'autenticatore è concepito per essere utilizzato una sola volta ed ha un tempo di validità molto breve. Il TGS può decifrare il ticket tramite la chiave che condivide con l'AS. Questo ticket indica che all'utente di C è stata fornita una chiave di sessione $K_{c,gs}$. Il TGS utilizza la chiave di sessione per decifrare l'autenticatore e può quindi confrontare il nome e l'indirizzo contenuti nell'autenticatore con quelli contenuti nel ticket e con l'indirizzo di rete del messaggio ricevuto. Se tutte le verifiche hanno successo, allora il TGS ha la certezza che colui che ha inviato il ticket è il suo legittimo proprietario. Si noti che il ticket non è uno strumento per provare l'identità ma è un mezzo per distribuire chiavi in modo sicuro. L'identità del client è provata dall'autenticatore. La minaccia che un avversario si impossessi del ticket sia dell'autenticatore al fine di utilizzarli in un momento successivo è scongiurata dal fatto che l'autenticatore può essere utilizzato una sola volta ed ha un tempo di validità molto limitato.

La risposta del TGS, nel messaggio (4), ha una struttura analoga al messaggio (2). Il messaggio è cifrato con la chiave di sessione condivisa dal TGS e da C, e contiene una chiave di sessione, che deve essere condivisa da C e dal server V, l'ID di V e il timestamp del ticket. Anche il ticket contiene la medesima chiave di sessione.

In questo modo C possiede un ticket di tipo service-granting riutilizzabile per V. Quando C presenta questo ticket, come mostrato nel messaggio (5), manda anche un autenticatore. Il server può decifrare il ticket, ricostruire la chiave di sessione e decifrare l'autenticatore.

Se è richiesta un'autenticazione reciproca, il server può rispondere con la modalità illustrata nel messaggio (6) della Tabella 4.1. Il server restituisce il valore del timestamp ottenuto dall'autenticatore, incrementato di 1 e cifrato con la chiave di sessione. C può decifrare questo messaggio per ricostruire il timestamp incrementato. C ha la certezza che il messaggio può essere stato generato solo da V, in quanto il messaggio è cifrato con la chiave di sessione. Il contenuto del messaggio assicura C del fatto che non si tratti del replay di una vecchia risposta.

Alla fine di questo processo, il client e il server condividono una chiave segreta che può essere utilizzata per cifrare messaggi futuri scambiati tra i due, o per scambiarsi una nuova chiave di sessione generata casualmente, per future comunicazioni.

La Tabella 4.2 riassume le motivazioni alla base di ogni elemento del protocollo di Kerberos, mentre la Figura 4.1 fornisce una versione semplificata delle interazioni.

Messaggio (1)	Il client richiede il ticket di tipo ticket-granting.
ID_c	Fornisce all'AS informazioni sull'identità dell'utente corrispondente al client.
ID_{tgs}	Serve all'AS per sapere che l'utente ha inoltrato una richiesta di accesso al TGS.
TS_1	Consente all'AS di verificare che l'orologio del client è sincronizzato con quello dell'AS.
Messaggio (2)	AS restituisce il ticket di tipo ticket-granting.
K_c	La cifratura si basa sulla password dell'utente, per fare in modo che l'AS ed il client siano in grado di verificare la password e di proteggere il contenuto del messaggio (2).
$K_{c,tgs}$	Copia della chiave di sessione accessibile al client; generata dall'AS per consentire scambi sicuri tra il client e il TGS senza che debbano condividere una chiave permanente.
ID_{tgs}	Conferma che il ticket è per il TGS.
TS_2	Fornisce al client informazioni sul tempo di generazione del ticket.
<i>Tempo di validità₂</i>	Fornisce al client informazioni sul tempo di validità del ticket.
$Ticket_{tgs}$	Ticket che il client deve utilizzare per accedere al TGS.

(a) Scambio relativo al servizio di autenticazione

Messaggio (3)	Il client richiede un ticket di tipo service-granting.
ID_v	Serve al TGS per essere a conoscenza del fatto che l'utente ha richiesto un accesso al server V.
$Ticket_{tgs}$	Assicura il TGS del fatto che l'utente sia stato autenticato dall'AS.
$Autenticatore_c$	Generato dal client per validare il ticket.
Messaggio (4)	Il TGS restituisce il ticket di tipo service-granting.
$K_{c,tgs}$	Chiave condivisa solo da C e da TGS; protegge il contenuto del messaggio (4).
$K_{c,v}$	Copia della chiave di sessione accessibile al client; generata dal TGS per rendere possibile uno scambio sicuro tra il client e il server senza la necessità che condividano una chiave permanente.
ID_v	Conferma che il ticket è per il server V.
TS_4	Fornisce al client informazioni sul tempo in cui il ticket è stato generato.
$Ticket_v$	Ticket che il client deve utilizzare per accedere al server V.
$Ticket_{tgs}$	Riutilizzabile in modo che l'utente non debba reimmettere la password.
K_{tgs}	Il ticket è cifrato con una chiave conosciuta solo dall'AS e dal TGS, al fine di evitare manomissioni.
$K_{c,tgs}$	Copia della chiave di sessione accessibile al TGS; utilizzata per decifrare l'autenticatore, e quindi per autenticare il ticket.
ID_c	Indica il proprietario legittimo del ticket.
AD_c	Impedisce l'utilizzo del ticket da parte di workstation diverse da quella che inizialmente ha richiesto il ticket.
ID_{tgs}	Assicura al server la corretta decifratura del ticket.
TS_2	Fornisce al TGS informazioni sul tempo di generazione del ticket.
<i>Tempo di validità₂</i>	Impedisce riutilizzi del ticket dopo la sua scadenza.

continua

Autenticatore_c	Assicura al TGS che colui che ha presentato il ticket è lo stesso client per cui tale ticket è stato emesso; l'autenticatore ha un tempo di validità molto limitato, per evitare attacchi di replay.
$K_{c,tgs}$	L'autenticatore è cifrato con una chiave conosciuta solo dal client e dal TGS, per evitare manomissioni.
ID_c	Affinché il ticket sia autenticato, deve essere uguale all'ID contenuto nel ticket.
AD_c	Affinché il ticket sia autenticato, deve essere uguale all'indirizzo contenuto nel ticket.
TS_3	Fornisce al TGS informazioni sul tempo di generazione dell'autenticatore.

(b) Scambio relativo al servizio di distribuzione

Messaggio (5)	Il client richiede un servizio.
$Ticket_v$	Assicura il server che l'utente è stato autenticato dall'AS.
$Autenticatore_c$	Generato dal client per validare il ticket.
Messaggio (6)	Autenticazione opzionale del server al client.
$K_{c,v}$	Assicura C del fatto che il messaggio proviene da V.
$TS_5 + 1$	Assicura C che non si tratta del riutilizzo (attacco di replay) di una vecchia risposta.
$Ticket_v$	Riutilizzabile in modo che il client non abbia bisogno di richiedere un nuovo ticket al TGS per ogni accesso allo stesso server.
K_v	Il ticket è cifrato con una chiave conosciuta solo dal TGS e dal server, al fine di evitare manomissioni.
$K_{c,v}$	Copia della chiave di sessione accessibile al client; utilizzata per decifrare l'autenticatore, e quindi per autenticare il ticket.
ID_c	Indica il proprietario legittimo del ticket.
AD_c	Evita l'utilizzo del ticket da workstation diverse da quella che inizialmente ha richiesto il ticket.
ID_v	Assicura al server la corretta decifratura del ticket.
TS_4	Fornisce al server informazioni sul tempo in cui il ticket è stato generato.
<i>Tempo di validità₄</i>	Impedisce attacchi di replay sul ticket dopo la sua scadenza.
$Autenticatore_c$	Assicura al server che colui che ha presentato il ticket è lo stesso client per cui tale ticket è stato generato; l'autenticatore ha un tempo di validità molto limitato, per evitare attacchi di replay.
$K_{c,v}$	L'autenticatore è cifrato con una chiave conosciuta solo dal client e dal server, per evitare manomissioni.
ID_c	Affinché il ticket sia autenticato, deve essere uguale all'ID contenuto nel ticket.
AD_c	Affinché il ticket sia autenticato, deve essere uguale all'indirizzo contenuto nel ticket.
TS_5	Fornisce al server informazioni sul tempo di generazione dell'autenticatore.

(c) Scambio relativo all'autenticazione client/server

Tabella 4.2 Motivazioni alla base degli elementi costitutivi del protocollo della versione 4 di Kerberos.

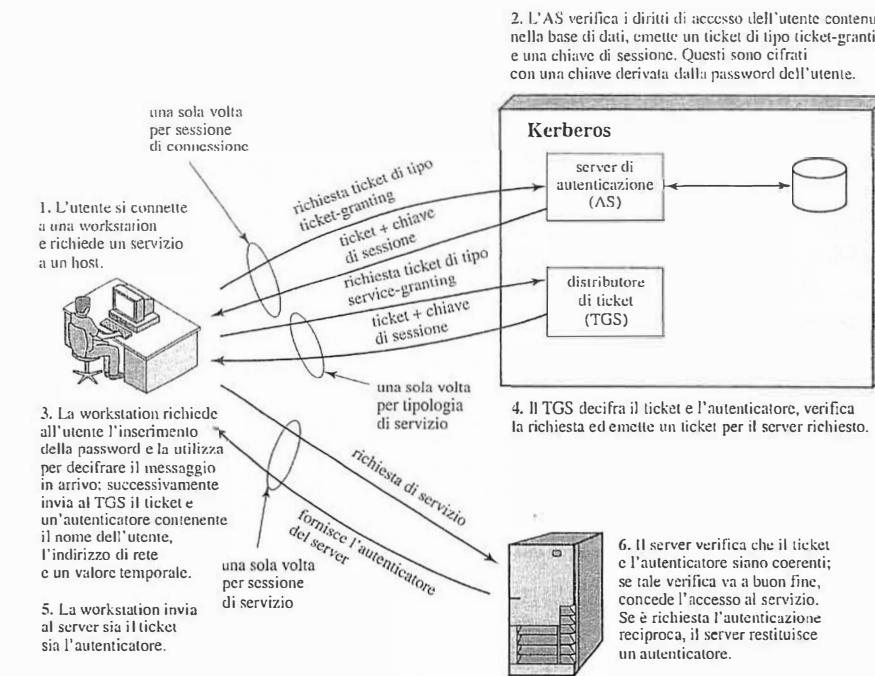


Figura 4.1 Panoramica di Kerberos.

Realm di Kerberos e Kerberos multiplo

Un ambiente Kerberos “comprendivo” (*full-service*), composto da un server Kerberos, da un certo numero di client e da un certo numero di server applicativi (*application server*), richiede i seguenti requisiti.

- Il server Kerberos deve avere nella sua base di dati l'ID utente (UID, *user ID*) e le password di tutti gli utenti partecipanti, codificati mediante una funzione hash. Tutti gli utenti sono registrati al server Kerberos.
- Il server Kerberos deve condividere una chiave segreta con ciascun server. Tutti i server sono registrati al server Kerberos.

Un ambiente di questo tipo viene chiamato **realm Kerberos**. Il concetto di *realm* può essere spiegato come segue. Un realm kerberos è un insieme di nodi amministrati che condividono lo stesso database Kerberos. Il database Kerberos risiede sul sistema di elaborazione master di Kerberos, che dovrebbe essere tenuto in una stanza fisicamente sicura. Una copia di sola lettura del database Kerberos potrebbe anche trovarsi su altri sistemi di elaborazione Kerberos. Tuttavia tutti i cambiamenti al database devono essere fatti sul sistema di elaborazione master. Il cambiamento o l'accesso ai contenuti di un database Kerberos richiedono la password master di Kerberos. Un concetto collegato è quello di **principal Kerberos**, che è un servizio o un utente che è noto al sistema Kerberos; ciascun principal Kerberos è identificato dal suo nome di tipo principal costituito da tre parti: un servizio o un nome utente, un nome di istanza e un nome di un realm.

In genere, reti di client e server che dipendono da differenti organizzazioni amministrative danno origine a realm distinti. Questo perché, di solito, non è possibile, o non è conforme alla politica amministrativa adottata, avere utenti e server appartenenti a uno stesso dominio amministrativo registrati presso server Kerberos diversi. Comunque, gli utenti appartenenti a un realm possono avere la necessità di accedere a server che appartengono ad altri realm, ed alcuni server potrebbero voler fornire servizi a utenti di altri realm, a patto che questi utenti siano autenticati.

Kerberos fornisce un meccanismo per consentire questa forma di autenticazione interrealm. Affinché due realm forniscano tale possibilità, è necessario un terzo requisito.

- Il server Kerberos, in ognuno dei realm coinvolti, condivide una chiave segreta con il server nell'altro realm. I due server Kerberos devono essere reciprocamente registrati.

Lo schema richiede che il server Kerberos in un realm si fidi del server Kerberos nell'altro realm per quanto riguarda l'autenticazione dei suoi utenti. Inoltre, i server partecipanti nel secondo realm devono anch'essi fidarsi del server Kerberos nel primo realm.

Una volta stabilite queste regole base, possiamo descrivere il meccanismo nel modo seguente (Figura 4.2). Un utente che desidera ottenere un servizio da un server collocato in un altro realm necessita di un ticket per quel server. Il client di tale utente segue la consueta procedura per ottenere l'accesso al TGS locale, e quindi richiede un ticket di tipo ticket-granting per un TGS remoto (cioè il TGS nell'altro realm). Il client può a questo punto richiedere al TGS remoto un ticket di tipo service-granting per il server cui vuole accedere nel realm del TGS remoto.

I dettagli degli scambi di messaggi illustrati nella Figura 4.2 sono i seguenti (si confronti con quanto illustrato nella Tabella 4.1):

- (1) $C \rightarrow AS: ID_C \| ID_{TGS} \| TS_1$
- (2) $AS \rightarrow C: E(K_c, [K_{c,TGS} \| ID_{TGS} \| TS_2 \| \text{Tempo di validità}_2 \| Ticket_{TGS}])$
- (3) $C \rightarrow TGS: ID_{TGS_{rem}} \| Ticket_{TGS} \| Autenticatore_c$
- (4) $TGS \rightarrow C: E(K_{c,TGS}, [K_{c,TGS_{rem}} \| ID_{TGS_{rem}} \| TS_4 \| Ticket_{TGS_{rem}}])$
- (5) $C \rightarrow TGS_{rem}: ID_{TGS_{rem}} \| Ticket_{TGS_{rem}} \| Autenticatore_c$
- (6) $TGS_{rem} \rightarrow C: E(K_{c,TGS_{rem}}, [K_{c,V_{rem}} \| ID_{V_{rem}} \| TS_6 \| Ticket_{V_{rem}}])$
- (7) $C \rightarrow V_{rem}: Ticket_{V_{rem}} \| Autenticatore_c$

Il ticket presentato al server remoto (V_{rem}) contiene informazioni sul realm in cui l'utente è stato inizialmente autenticato. Tale server decide se soddisfare o meno la richiesta remota.

Quando il numero di realm aumenta considerevolmente, l'approccio appena illustrato diventa inefficiente. Infatti, con N realm sono necessari $N(N - 1)/2$ scambi sicuri di chiavi affinché ogni realm Kerberos possa interagire con tutti gli altri.

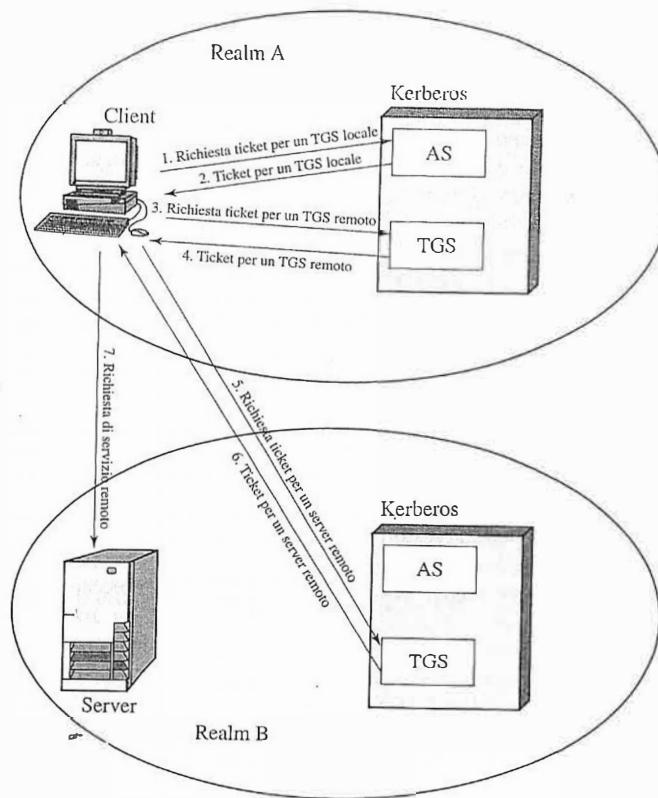


Figura 4.2 Richiesta di servizio in un altro realm.

Versione 5 di Kerberos

La versione 5 di Kerberos è specificata nel documento RFC 1510 e contiene considerevoli migliorie rispetto alla versione 4 [KOHL94]. La descrizione della versione 5 di Kerberos inizierà con una panoramica delle modifiche rispetto alla versione 4. Successivamente verrà illustrato il protocollo della versione 5.

Differenze tra la versione 4 e la versione 5

La versione 5 è concepita per superare i limiti della versione 4 in due aree: difetti ambientali e limiti tecnici. Vediamo brevemente le migliorie apportate in ciascuna area⁶.

La versione 4 di Kerberos fu sviluppata per un utilizzo nell'ambiente del progetto Athena e, per tale motivo, non soddisfaceva pienamente il requisito di essere di utilizzo più generale. Questa caratteristica ha comportato i seguenti difetti ambientali.

- Dipendenza dal sistema di cifratura.** La versione 4 presuppone l'utilizzo di DES e quindi soffre dei problemi legati alle restrizioni sull'esportazione di DES e alla sua robustezza. Nella versione 5, il testo cifrato è contrassegnato con un identificatore del tipo di cifratura; ciò consente di utilizzare una qualsiasi tecnica di cifratura. Le chiavi di cifratura sono contrassegnate con un tipo e una lunghezza, in modo da consentire l'utilizzo della stessa chiave in algoritmi diversi e di specificare diverse varianti di un certo algoritmo.
- Dipendenza dal protocollo Internet.** La versione 4 richiede l'utilizzo degli indirizzi IP (*Internet Protocol*). Altri tipi di indirizzi come, ad esempio, l'indirizzo di rete ISO, non sono consentiti. Nella versione 5, gli indirizzi di rete sono contrassegnati con il tipo e la lunghezza, in modo da consentire l'utilizzo di un qualsiasi tipo di indirizzo di rete.
- Ordinamento dei byte del messaggio.** Nella versione 4, il mittente di un messaggio utilizza un ordinamento dei byte di sua scelta e contrassegna il messaggio per indicare se all'indirizzo più basso corrisponde il byte meno significativo o quello più significativo. Questa tecnica funziona, ma non segue le convenzioni stabilite. Nella versione 5, tutte le strutture dei messaggi sono specificate tramite ASN.1 (*abstract syntax notation one*) e BER (*basic encoding rule*), che forniscono un ordinamento non ambiguo dei byte.
- Tempo di validità del ticket.** I valori di tale parametro nella versione 4 erano codificati in 8 bit in unità di cinque minuti ciascuna. Questo implica che il massimo tempo di validità che può essere specificato è $2^8 \times 5 = 1280$ minuti, cioè poco più di 21 ore. Tale limite può non essere adeguato per alcune applicazioni (per esempio, simulazioni di lunga durata che hanno necessità di credenziali Kerberos valide durante la loro esecuzione). Nella versione 5, il ticket include esplicitamente un tempo di inizio e di fine, rendendo quindi possibile la specifica di ticket con un tempo di validità arbitrario.
- Inoltro dell'autenticazione.** La versione 4 non consente di inoltrare credenziali generate per un certo client ad altri host per un loro utilizzo da parte di altri client. Questa possibilità consentirebbe a un client di accedere a un server ed al server di accedere un altro server per conto del client. Ad esempio, si supponga che un client invii una richiesta a un server di stampa che accede quindi al file del client tramite un file server, utilizzando le credenziali del client per l'accesso. La versione 5 fornisce tale possibilità.
- Autenticazione inter-realm.** Come descritto in precedenza, nella versione 4 l'interoperabilità tra N realm si basa sull'ordinamento di N^2 relazioni tra server Kerberos. La versione 5 fornisce un metodo che richiede un minor numero di relazioni.

Oltre a queste limitazioni connesse con l'ambiente, la versione 4 del protocollo presenta alcuni **limiti tecnici** documentati, per la maggior parte, in [BELL90]; limiti che la versione 5 si propone di superare.

⁶ La discussione successiva si basa sulla trattazione in [KOHL94].

- Doppia cifratura.** Si noti dalla Tabella 4.1 [messaggi (2) e (4)] che i ticket forniti ai client sono cifrati due volte, una volta con la chiave segreta del server di riferimento e successivamente con una chiave segreta conosciuta dal client. La seconda cifratura non è necessaria e costituisce uno spreco dal punto di vista computazionale.
- Cifratura PCBC.** Nella versione 4, la cifratura si basa su una modalità non-standard di DES conosciuta come modalità *propagating cipher block chaining* (PCBC)⁷. È stato provato che tale modalità è vulnerabile a un attacco che coinvolge l'interscambio di blocchi di testo cifrati [KOHL89]. L'intento di PCBC è quello di fornire una verifica di integrità come parte delle operazioni di cifratura. La versione 5 fornisce meccanismi di integrità esplicativi permettendo, quindi, di utilizzare la modalità standard CBC per la cifratura. In particolare, viene allegata al messaggio, prima che venga cifrato usando CBC, una somma di controllo (*checksum*) o un codice hash.
- Chiavi di sessione.** Ciascun ticket contiene una chiave di sessione utilizzata dal client per cifrare l'autenticatore inviato al servizio per cui il ticket è stato generato. Inoltre, la chiave di sessione può essere utilizzata dal client e dal server in tempi successivi per proteggere i messaggi durante quella sessione. Comunque, dal momento che lo stesso ticket può essere ripetutamente utilizzato per ottenere servizi da un certo server, esiste il rischio che un avversario riutilizzi messaggi appartenenti a una vecchia sessione e li invii al client o al server. Nella versione 5, un client e un server possono accordarsi su una chiave di sottosessione, da utilizzare per un'unica connessione. Un nuovo accesso da parte del client avrebbe come risultato l'utilizzo di una nuova chiave di sottosessione.
- Attacchi alle password.** Entrambe le versioni sono vulnerabili a questo tipo di attacchi. Il messaggio inviato dall'AS al client contiene informazioni che sono cifrate con una chiave basata sulla password del client⁸. Un avversario può intercettare questo messaggio e tentare di decifrarlo provando diverse password. Se il risultato di uno di questi tentativi ha la forma corretta, significa che l'avversario è entrato in possesso della password del client e può successivamente utilizzarla per ottenere da Kerberos credenziali di autenticazione. Questo tipo di attacco alle password è analogo a quello che verrà descritto nel Capitolo 9 e, quindi, sono applicabili le stesse contromisure. La versione 5 fornisce un meccanismo conosciuto come pre-autenticazione, che dovrebbe rendere più difficili gli attacchi alle password, ma non è tuttavia in grado di prevenirli.

Dialogo di autenticazione della versione 5

La Tabella 4.3 sintetizza il dialogo base della versione 5. Questo dialogo può essere spiegato più chiaramente operando un confronto con la versione 4 (Tabella 4.1).

Si consideri in primo luogo lo **scambio relativo al servizio di autenticazione**. Il messaggio (1) è una richiesta di un ticket di tipo ticket-granting da parte del client. Come nella versione 4, il messaggio contiene l'ID dell'utente e del TGS. Sono stati però aggiunti i seguenti, nuovi elementi.

⁷ Tale modalità verrà descritta nell'Appendice 4A.

⁸ L'Appendice 4A descrive la trasformazione delle password in chiavi di cifratura.

```
(1) C → AS Opzioni ||| IDc ||| Realmc ||| IDqs ||| Tempi ||| Nonce1
(2) AS → C Realmc ||| IDC ||| Ticketqs ||| E(Kc,qs, [Kc,qs] ||| Tempi ||| Nonce1 ||| Realmqs ||| IDqs)
```

$$\text{Ticket}_{qs} = E(K_{c,qs}, [\text{Flag} ||| K_{c,qs}] ||| \text{Realm}_c ||| ID_c ||| AD_c ||| \text{Tempi})$$

(a) Scambio relativo al servizio di autenticazione: per ottenere un ticket di tipo ticket-granting

```
(3) C → TGS Opzioni ||| IDc ||| Tempi ||| Nonce2 ||| Ticketqs ||| Autenticatorec
(4) TGS → C Realmc ||| IDC ||| Ticketv ||| E(Kc,v, [Kc,v] ||| Tempi ||| Nonce2 ||| Realmv ||| IDv)
```

$$\text{Ticket}_v = E(K_{c,v}, [\text{Flag} ||| K_{c,v}] ||| \text{Realm}_c ||| ID_c ||| AD_c ||| \text{Tempi})$$

$$\text{Autenticatore}_c = E(K_{c,qs}, [ID_c ||| \text{Realm}_c ||| TS_1])$$

(b) Scambio relativo al servizio di distribuzione di ticket: per ottenere un ticket di tipo service-granting

```
(5) C → V Opzioni ||| Ticketv ||| Autenticatorec
(6) V → C E(Kc,v, TS2 ||| Sottochiave ||| Seq#)
```

$$\text{Ticket}_v = E(K_{c,v}, [\text{Flag} ||| K_{c,v}] ||| \text{Realm}_c ||| ID_c ||| AD_c ||| \text{Tempi})$$

$$\text{Autenticatore}_c = E(K_{c,qs}, [ID_c ||| \text{Realm}_c ||| TS_2 ||| \text{Sottochiave} ||| \text{Seq}\#])$$

(c) Scambio relativo all'autenticazione client/server: per ottenere un servizio

Tabella 4.3 Sintesi dello scambio di messaggi nella versione 5 di Kerberos.

- **Realm:** indica il realm dell'utente.
- **Opzioni:** elemento utilizzato per richiedere un'opportuna configurazione dei flag nel ticket restituito.
- **Tempi:** elemento utilizzato dal client per richiedere che nel ticket siano impostati i seguenti parametri temporali:
 - from: il tempo di inizio desiderato per il ticket richiesto;
 - till: il tempo di scadenza per il ticket richiesto;
 - rtime: tempo di rinnovo richiesto.
- **Nonce:** valore casuale da ripetere nel messaggio (2) per assicurare che la risposta sia recente e non sia stata riutilizzata da un avversario.

Il messaggio (2) restituisce: un ticket di tipo ticket-granting, informazioni necessarie per l'identificazione del client e un blocco cifrato mediante la chiave di cifratura basata sulla password dell'utente. Questo blocco contiene: la chiave di sessione da utilizzare tra il client e il TGS, i tempi specificati nel messaggio (1), il nonce contenuto nel messaggio (1) e le informazioni necessarie per identificare il TGS. A sua volta il ticket contiene: la chiave di sessione, informazioni per accettare l'identità del client, i tempi richiesti e flag che riflettono lo stato del ticket e le opzioni richieste. I flag introducono nella versione 5 nuove funzionalità significative. Per il momento, la discussione di tali flag viene posticipata per concentrarsi sulla struttura generale del protocollo della versione 5.

Confrontiamo ora gli **scambi relativi al servizio di distribuzione dei ticket** delle versioni 4 e 5. Si può notare che il messaggio (3) contiene, in entrambe le versioni, un autenticatore, un ticket e il nome del servizio richiesto. In aggiunta, la versione 5 contiene i tempi richiesti, le opzioni per il ticket e un nonce, tutti con funzioni simili a quelle del messaggio (1). L'autenticatore è essenzialmente lo stesso di quello utilizzato nella versione 4.

Il messaggio (4) ha la stessa struttura del messaggio (2) e restituisce un ticket e le informazioni necessarie al client, queste ultime cifrate con la chiave di sessione che è al momento condivisa da client e TGS.

Infine, la versione 5 contiene numerose nuove funzionalità per quanto riguarda lo scambio relativo all'autenticazione tra client e server. Nel messaggio (5), il client può richiedere come opzione che sia necessaria l'autenticazione reciproca. L'autenticatore contiene alcuni campi aggiuntivi.

- **Sottochiave.** Indica la chiave di cifratura scelta dal client per proteggere la sessione applicativa corrente. Se tale campo viene omesso, si utilizza la chiave di sessione contenuta nel ticket ($K_{c,s}$).
- **Numero di sequenza.** È un campo opzionale usato per specificare il numero di sequenza iniziale che il server deve utilizzare per i messaggi inviati al client durante la sessione corrente. I messaggi possono contenere un numero di sequenza per smascherare eventuali attacchi di replay.

Se è richiesta l'autenticazione reciproca, il server risponde con il messaggio (6) che contiene il timestamp presente nell'autenticatore. Nella versione 4, il timestamp è incrementato di un'unità. Tale incremento non è necessario nella versione 5 in quanto il formato del messaggio non rende possibile a un avversario la generazione del messaggio (6) senza la conoscenza delle appropriate chiavi di cifratura. Il campo che indica la sottochiave, se non è omesso, sovrascrive l'analogo campo eventualmente contenuto nel messaggio (5). Il campo opzionale numero di sequenza specifica il numero di sequenza iniziale che deve essere utilizzato dal client.

Flag del ticket

Il campo contenente flag nei ticket della versione 5 fornisce funzionalità addizionali rispetto a quelle fornite dalla versione 4. La Tabella 4.4 riassume i flag che possono comparire in un ticket.

Il flag INITIAL specifica che il ticket in questione non è stato generato dal TGS ma dall'AS. Quando un client richiede al TGS un ticket di tipo service-granting, presenta un ticket di tipo ticket-granting ottenuto dall'AS. Nella versione 4, questo era l'unico modo per ottenere un ticket di tipo service-granting. La versione 5 fornisce un'ulteriore possibilità, cioè che il client possa ottenere un ticket di tipo service-granting direttamente dall'AS. Il vantaggio è che un server, come ad esempio un server addetto alla modifica delle password, potrebbe essere interessato a conoscere se la password di un certo client è stata recentemente verificata.

Il flag PRE-AUTHENT, se inizializzato, indica che quando l'AS ha ricevuto la richiesta iniziale – messaggio (1) – ha autenticato il client prima di generare un ticket. Non è però specificato il tipo di pre-autenticazione utilizzata. Ad esempio, l'implementazione

INITIAL	Il ticket è stato emesso utilizzando il protocollo dell'AS e non sulla base di un ticket di tipo ticket-granting.
PRE-AUTHENT	Durante l'autenticazione iniziale, il client è stato autenticato KDC prima dell'emissione di un ticket.
HW-AUTHENT	Il protocollo utilizzato per l'autenticazione iniziale ha richiesto l'utilizzo di hardware che ci si aspetta sia in possesso solo del client in questione.
RENEWABLE	Indica al TGS che il ticket può essere utilizzato per ottenere un ticket sostitutivo con un tempo di validità più elevato.
MAY-POSTDATE	Indica al TGS che un ticket postdatato può essere emesso sulla base del ticket di tipo ticket-granting in questione.
POSTDATED	Indica che il ticket è stato postdatato; il server finale può verificare il campo authTime per avere informazioni sul tempo in cui è stata effettuata l'autenticazione iniziale.
INVALID	Il ticket non è valido e deve essere validato da KDC prima di poter essere utilizzato.
PROXiable	Indica al TGS che, sulla base del ticket presentato, può essere generato un nuovo ticket di tipo service-granting con un diverso indirizzo di rete.
PROXY	Indica che il ticket è un proxy.
FORWARDABLE	Indica al TGS che, sulla base del ticket di tipo ticket-granting corrente, può essere generato un nuovo ticket di tipo ticket-granting con un diverso indirizzo di rete.
FORWARDED	Indica che il ticket è stato inoltrato oppure è stato emesso sulla base di un'autenticazione che ha coinvolto un ticket di tipo ticket-granting inoltrato.

Tabella 4.4 Flag della versione 5 di Kerberos.

della versione 5 fornita dal MIT esegue di default una pre-autenticazione basata su timestamp cifrato. Quando un utente desidera ricevere un ticket, deve inviare all'AS un blocco di pre-autenticazione contenente un valore casuale, un numero di versione e un timestamp. Tutte queste informazioni sono cifrate con la chiave basata sulla password del client. AS decifra il blocco e invia un ticket di tipo ticket-granting solo se il timestamp nel blocco di pre-autenticazione è contenuto nell'intervallo di tolleranza consentito (definito per tener conto della velocità dell'orologio e dei ritardi di rete). Un'altra opzione è l'utilizzo di una smart card che genera password che cambiano continuamente e che sono incluse nel messaggio di pre-autenticazione. Le password generate dalla smart card possono essere basate sulla password utente ma sono trasformate dalla smart card in modo da essere equivalenti a password generate arbitrariamente. Questo tipo di approccio previene un attacco basato su password facilmente indovinabili. Il flag HW-AUTHENT indica l'utilizzo di una smart card o di un dispositivo analogo.

Se un ticket ha un tempo di validità elevato, sussiste il rischio che possa essere rubato da un avversario che lo può utilizzare per un lungo periodo. Se per ovviare a tale minaccia viene utilizzato un tempo di validità più corto, si deve tenere in considerazione un costo aggiuntivo in quanto deve essere acquisito un numero maggiore di ticket. Nel caso di un ticket di tipo ticket-granting, il client dovrebbe memorizzare la chiave segreta dell'utente, cosa che chiaramente costituisce un rischio, oppure chiedere ripetutamente la password all'utente. Un compromesso è costituito dall'utilizzo di ticket rinnovabili. Un ticket in cui il flag RENEWABLE è non nullo contiene due diversi tempi di fine validità: uno per il ticket in questione e l'altro che è il valore massimo consentito come tempo di validità. Un client può avere un rinnovo del ticket presentandolo al TGS unitamente alla richiesta di un nuovo tempo di fine validità. Se il nuovo tempo è minore del massimo tempo consentito, il TGS può generare un nuovo ticket con un nuovo tempo di sessione e un tempo di fine validità più elevato. Il vantaggio di questo meccanismo è che il TGS può rifiutarsi di rinnovare un ticket che è stato indicato come rubato.

Un client può richiedere che l'AS fornisca un ticket di tipo ticket-granting con il flag MAY-POSTDATE inizializzato. Il client può quindi utilizzare questo ticket per richiedere un ulteriore ticket che è indicato dal TGS come POSTDATED e INVALID. Successivamente, il client può sottoporre il ticket postdatato per la validazione. Questo modo di procedere può essere utile per l'esecuzione di lunghi job in modalità batch su un server che periodicamente richiede un ticket. Il client può ottenere in una sola volta un certo numero di ticket per la sessione, con tempi distribuiti in un certo intervallo. All'inizio tutti i ticket, ad eccezione del primo, non sono validi. Quando, durante l'esecuzione, si raggiunge un punto in cui è richiesto un nuovo ticket, il client può ottenere la validazione del ticket appropriato. Con questo approccio, il client non deve ripetutamente utilizzare il suo ticket di tipo ticket-granting per ottenere un ticket di tipo service-granting.

Nella versione 5 è possibile che un server operi come proxy per conto di un client, ereditando dal client le sue credenziali e i suoi privilegi, per richiedere un servizio a un altro server. Se un client vuole utilizzare questo meccanismo, richiede un ticket di tipo ticket-granting con il flag PROXIABLE inizializzato. Quando il ticket è presentato al TGS, questi può generare un ticket di tipo service-granting con un indirizzo di rete diverso; quest'ultimo ticket avrà il flag PROXY inizializzato. Un'applicazione che riceve questo ticket può accettarlo oppure richiedere un'ulteriore autenticazione per fornire una traccia di audit⁹.

Il concetto di proxy è un caso particolare della più potente procedura di inoltro (*forwarding*). Se un ticket contiene il flag FORWARDABLE inizializzato, un TGS può generare per il richiedente un ticket di tipo ticket-granting con un indirizzo di rete diverso e il flag FORWARDED inizializzato. Questo ticket può essere presentato a un TGS remoto. Questa capacità consente a un client di ottenere l'accesso a un server in un altro realm senza la necessità che ciascun server Kerberos condivida una chiave segreta con i server Kerberos in ciascuno degli altri realm. Ad esempio, i realm potrebbero essere organizzati gerarchicamente. Il client potrebbe risalire l'albero fino ad arrivare a un nodo comune, per poi discenderlo fino a raggiungere il realm desiderato. Ogni passo del cammino richiederebbe l'inoltro di un ticket di tipo ticket-granting al successivo TGS presente sul cammino.

⁹ Per una trattazione sui proxy si veda [NEUM93b].

4.2 Servizio di autenticazione X.509

La raccomandazione ITU-T X.509 è parte della serie di raccomandazioni X.500 volte alla definizione di un servizio di directory. In linea generale, una directory è un server o un insieme distribuito di server che mantengono una base di dati contenente informazioni sugli utenti, tra cui quelle relative alla corrispondenza tra nomi utente e indirizzi di rete, come pure altri attributi e informazioni sugli utenti.

X.509 definisce un'architettura di riferimento per l'erogazione di servizi di autenticazione da parte della directory X.500 ai suoi utenti. La directory può esser utilizzata come luogo in cui mantenere i certificati a chiave pubblica dei quelli trattati nel Capitolo 3. Ciascun certificato contiene la chiave pubblica di un utente ed è firmato con la chiave privata di un'autorità certificativa (CA, certification authority) fidata. Inoltre, X.509 definisce protocolli di autenticazione alternativi basati sull'utilizzo di certificati a chiave pubblica.

L'importanza di questo standard è dovuta al fatto che la struttura dei certificati e dei protocolli di autenticazione definiti in X.509 sono utilizzati in molteplici contesti. Ad esempio, il formato di certificato X.509 è utilizzato in: S/MIME (Capitolo 5), Sicurezza IP (Capitolo 6), e SSL/TLS e SET (Capitolo 7).

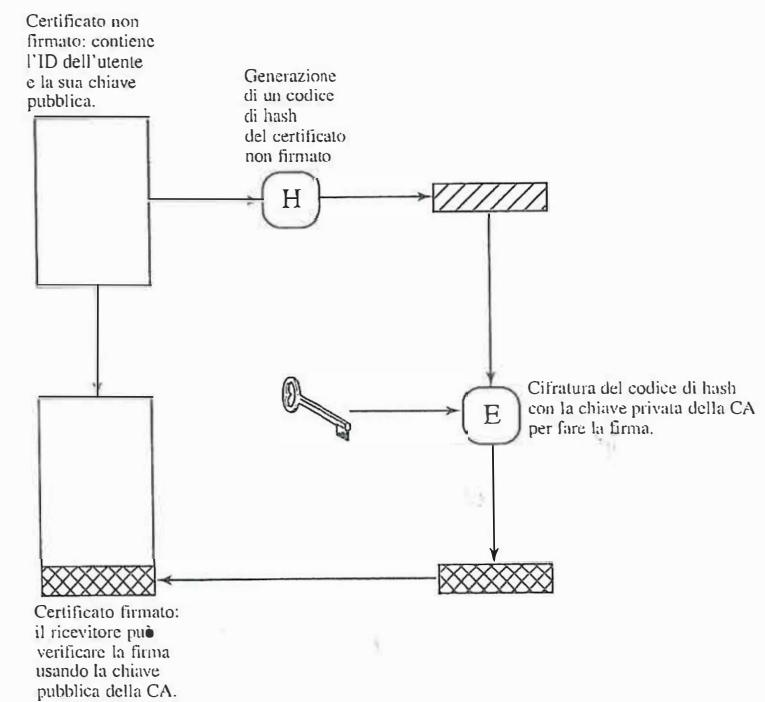


Figura 4.3 Uso di un certificato a chiave pubblica.

La prima versione di X.509 fu emanata nel 1988. Lo standard fu successivamente revisionato per rispondere ad alcune problematiche di sicurezza documentate in [IANS90] e [MITC90]; nel 1993 è stata emessa una raccomandazione revisionata. Una terza versione fu redatta nel 1995 e revisionata nel 2000.

X.509 si basa sull'utilizzo della crittografia a chiave pubblica e delle firme digitali. Lo standard non impone l'utilizzo di un particolare algoritmo di cifratura, ma raccomanda RSA. Lo schema di firma digitale ipotizza l'utilizzo di una funzione hash. Anche in questo caso lo standard non impone alcun particolare algoritmo hash. La raccomandazione del 1988 conteneva la descrizione di un algoritmo hash consigliato, eliminato però dalla raccomandazione del 1993 in quanto non sicuro. La Figura 4.3, a pagina precedente, illustra come viene generato un certificato a chiave pubblica.

Certificati

Il fulcro dello schema X.509 è costituito dai certificati a chiave pubblica associati a ciascun utente. Si assume che i certificati utente siano creati da un'autorità certificativa (CA) fidata e posti nella directory dalla CA o dall'utente. Non è compito del directory server creare le chiavi pubbliche o effettuare la certificazione; il suo compito è unicamente quello di fornire un luogo facilmente accessibile da cui gli utenti possono ottenere i certificati.

La Figura 4.4a illustra il formato generale di un certificato in cui sono riconoscibili i seguenti elementi:

- Versione.** Serve per attuare una distinzione tra le versioni successive del formato del certificato; la versione di default è la 1. Se sono presenti gli elementi identificatore unico di chi emette il certificato e identificatore unico del soggetto, allora questo elemento deve avere come valore la versione 2. Se sono presenti una o più estensioni allora la versione deve essere la 3.
- Numero di serie.** Valore intero, unico per la CA che emette il certificato, associato in modo non ambiguo al certificato in questione.
- Identificatore dell'algoritmo di firma.** Identificatore dell'algoritmo utilizzato per firmare il certificato, con gli eventuali parametri associati. Dal momento che questa informazione è ripetuta nel campo Firma alla fine del certificato, l'utilità di questo campo è pressoché nulla.
- Nome di chi emette.** Nome X.500 della CA che ha creato e firmato il certificato.
- Periodo di validità.** Campo costituito da due date corrispondenti agli estremi del periodo di validità del certificato.
- Nome del soggetto.** Nome dell'utente cui il certificato si riferisce; vale a dire che quest'ultimo certifica la chiave pubblica del soggetto che detiene la corrispondente chiave privata.
- Informazioni sulla chiave pubblica del soggetto.** Contiene la chiave pubblica del soggetto e un identificatore dell'algoritmo con cui tale chiave deve essere utilizzata, insieme a ogni eventuale parametro.
- Identificatore unico di chi emette il certificato.** Campo opzionale, costituito da una stringa di bit, utilizzato per identificare univocamente la CA che emette il certificato, nel caso in cui il nome X.500 sia stato riutilizzato per entità differenti.

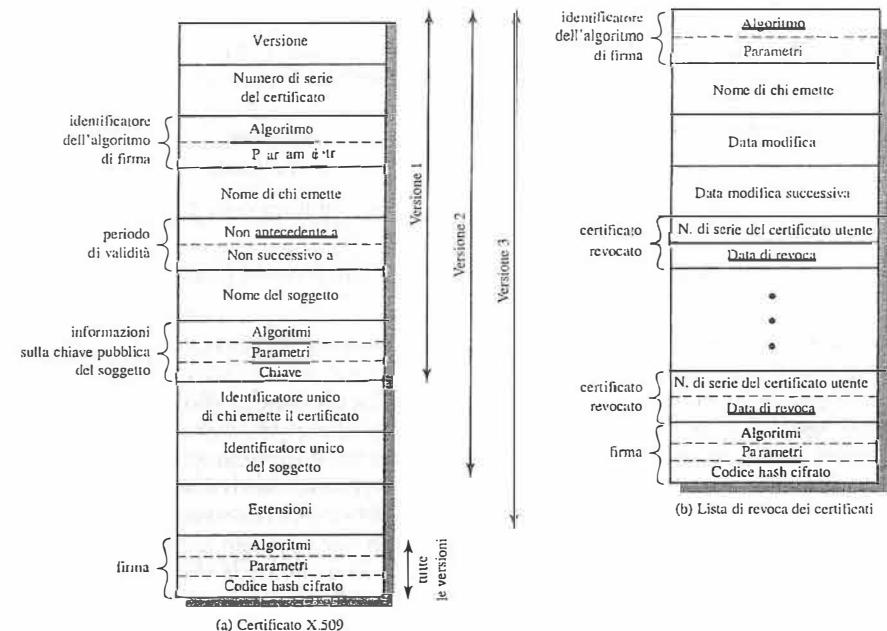


Figura 4.4 Formati X.509.

- Identificatore unico del soggetto.** Campo opzionale costituito da una stringa di bit utilizzato per identificare univocamente il soggetto nel caso in cui il nome X.500 sia stato riutilizzato per entità differenti.
- Estensioni.** Insieme di uno o più campi estensione. Le estensioni sono state aggiunte nella versione 3 e sono trattate nel seguito di questo paragrafo.
- Firma.** Copre tutti gli altri campi del certificato; contiene il codice hash degli altri campi, cifrato con la chiave privata della CA. Tale campo contiene l'identificatore dell'algoritmo utilizzato per la firma.

I campi che mantengono informazioni sugli identificatori unici furono aggiunti nelle versioni 2 per gestire il possibile riutilizzo, nel corso del tempo, dei nomi del soggetto e/o della CA che emana il certificato. Questi campi sono raramente utilizzati nella pratica.

Per specificare un certificato, lo standard utilizza la seguente notazione:

CA <> A >> CA {V, SN, AI, CA, T_A, A, Ap}

dove

Y <> X = certificato dell'utente X emesso dall'autorità certificativa Y

Y {I} = firma di I da parte di Y. È composta da I con accodato un codice hash cifrato.

La CA firma il certificato con la sua chiave privata. Se un utente conosce la corrispondente chiave pubblica, allora l'utente può verificare la validità dei certificati firmati dalla CA. Questo rappresenta il tipico schema di firma digitale precedentemente illustrato nella Figura 3.2b.

Come ottenere un certificato utente

I certificati utente generati da una CA hanno le seguenti caratteristiche.

- Un qualsiasi utente che abbia accesso alla chiave pubblica della CA può ricostruire la chiave pubblica dell'utente cui il certificato si riferisce.
- Nessun'altra parte, ad eccezione della CA, può modificare il certificato senza essere scoperta.

Dato che i certificati non sono falsificabili, possono essere messi in una directory senza adottare particolari precauzioni per la loro protezione.

Se tutti gli utenti si iscrivono alla stessa CA, significa che esiste una fiducia comune verso questa. Tutti i certificati utente possono essere posti nella directory per essere accessibili da tutti gli utenti. Inoltre, in questo modo un utente può trasmettere direttamente ad altri utenti il suo certificato. In entrambi i casi, una volta entrato in possesso del certificato di A, B avrà la certezza che i messaggi da lui cifrati con la chiave pubblica di A risulteranno sicuri da intercettazioni e i messaggi firmati con la chiave privata di A non potranno essere contraffatti.

Se il numero di utenti è elevato, potrebbe non essere possibile che tutti gli utenti si iscrivano alla stessa CA. Dal momento che è compito della CA firmare i certificati, ciascun utente dovrebbe avere una copia della chiave pubblica della CA al fine di verificare l'autenticità delle firme. Questa chiave pubblica deve essere fornita a tutti gli utenti con una modalità completamente sicura (dal punto di vista sia dell'autenticità sia dell'integrità) in modo tale che l'utente si fidi del certificato associato. Per questo, quando si ha a che fare con un elevato numero di utenti, può essere più indicata una soluzione in cui un certo numero di CA fornisce con modalità sicura la sua chiave pubblica a una parte degli utenti.

Si ipotizzi che A abbia ottenuto un certificato dall'autorità certificativa X_1 , e che B abbia ottenuto un certificato dalla CA X_2 . Se A non è a conoscenza, tramite una modalità sicura, della chiave pubblica di X_2 , il certificato di B, generato da X_2 , non è di alcuna utilità per A. A può leggere il certificato di B, ma non può verificarne la firma. Se però le CA coinvolte si sono scambiate, tramite una modalità sicura, le loro chiavi pubbliche, allora la procedura di seguito illustrata può essere utilizzata da A per ottenere la chiave pubblica di B.

1. A ottiene, dalla directory, il certificato di X_2 firmato da X_1 . Dal momento che A ha acquisito la chiave pubblica di X_1 tramite una modalità sicura, A può ottenere la chiave pubblica di X_2 dal suo certificato e verificarne l'autenticità tramite la firma di X_1 apposta sul certificato.
2. A accede nuovamente alla directory e ottiene il certificato di B, firmato da X_2 . Dal momento che A è ora in possesso di una copia fidata della chiave pubblica di X_2 , può verificarne la firma e ottenere in modo sicuro la chiave pubblica di B.

A ha utilizzato una sequenza di certificati per entrare in possesso della chiave pubblica di B. Utilizzando la notazione di X.509, la sequenza di certificati può essere espressa come segue:

$X_1 <<X_2>> X_2 <>$

Analogamente, B può ottenere la chiave pubblica di A utilizzando la sequenza inversa:

$X_2 <<X_1>> X_1 <<A>>$

L'utilizzo di questo schema non è limitato a sequenze di due certificati. Un percorso di CA di lunghezza arbitraria può essere utilizzato per produrre una sequenza. Una sequenza di N elementi è espressa tramite la notazione:

$X_1 <<X_2>> X_2 <<X_3>> \dots X_N <>$

In questo caso, ciascuna CA della coppia (X_i, X_{i+1}) nella sequenza deve aver creato i certificati per l'altra.

Tutti questi certificati generati dalle CA per altre CA devono apparire nella directory, e l'utente deve conoscere le modalità con cui sono collegati per poter seguire il percorso che lo conduce al certificato a chiave pubblica di un altro utente. X.509 consiglia di strutturare le CA in una gerarchia in modo da rendere agevole la navigazione.

La Figura 4.5, tratta da X.509, mostra un esempio di gerarchia. I nodi connessi rappresentano la relazione gerarchica che sussiste tra le CA; i riquadri associati ai nodi indicano i certificati mantenuti nella directory per ogni CA. L'elemento nella directory corrispondente a ciascuna CA contiene due tipi di certificati:

- **certificati di tipo forward:** certificati di X generati da altre CA;
- **certificati di tipo reverse:** certificati per altre CA generati da X.

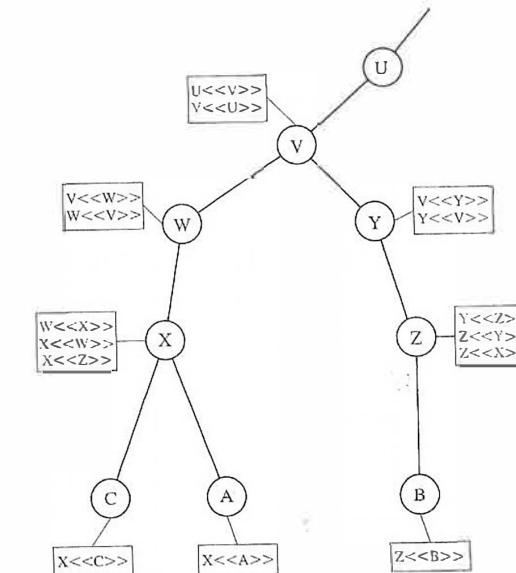


Figura 4.5 Esempio di gerarchia X.509 di CA .

Nell'esempio che segue, l'utente A, per stabilire un percorso di certificazione verso B, può ottenere dalla directory i seguenti certificati:

X <>W>> W <>V>> V <>Y>> Y <>Z>> Z <>B>>

Una volta che A ha ottenuto tali certificati, può ripercorrere il percorso di certificazione in senso contrario per ricostruire una copia fidata della chiave pubblica di B. Con questa chiave pubblica, A può inviare messaggi cifrati a B. Se A desidera, a sua volta, ricevere messaggi cifrati da B, o firmare messaggi inviati a B, allora B richiederà la chiave pubblica di A, che può essere ottenuta mediante il seguente cammino di certificazione:

Z <>Y>> Y <>V>> V <>W>> W <>X>> X <>A>>

B può ottenere questa sequenza di certificati dalla directory, oppure A può fornirli a B come parte del messaggio iniziale.

Revoca dei certificati

Come evidenziato nella Figura 4.4, ogni certificato include un periodo di validità. Solitamente, viene emesso un nuovo certificato poco prima della scadenza di quello vecchio. Inoltre, a volte, può essere utile revocare un certificato prima della sua scadenza per una delle seguenti ragioni:

1. si presume che la chiave privata dell'utente sia compromessa;
2. l'utente non è più certificato dalla CA del certificato in questione;
3. si presume che il certificato della CA sia compromesso.

Ciascuna CA deve mantenere una lista di tutti i certificati che ha generato e che sono stati revocati ma non sono ancora scaduti. Tale lista deve includere i certificati generati sia per utenti sia per altre CA. Anche questa lista deve essere resa disponibile nella directory.

Ogni lista di revoca dei certificati (CRL, *certificate revocation list*) resa disponibile nella directory è firmata dalla CA che la emette e contiene (Figura 4.4b): il nome della CA che l'ha emessa, la data di creazione della lista, la data prevista per la pubblicazione della CRL successiva, una voce per ogni certificato revocato. Ogni voce contiene il numero di serie di un certificato e la sua data di revoca. Dal momento che i numeri di serie sono unici per una CA, tali numeri sono sufficienti per identificare in modo univoco un certificato.

Quando un utente riceve un certificato tramite un messaggio, deve essere in grado di capire se si tratta di un certificato revocato. Per questo l'utente potrebbe controllare la directory ogni volta che riceve un certificato. Per evitare ritardi (ed eventuali costi) connessi alla ricerca nella directory, l'utente potrebbe mantenere una cache locale contenente i certificati e le liste dei certificati revocati.

Procedure di autenticazione

X.509 contiene anche tre procedure alternative di autenticazione da utilizzare a seconda dei contesti applicativi. Tutte queste procedure utilizzano meccanismi di firma a chiave pubblica. Si ipotizza che ciascuna delle due parti coinvolte conosca la chiave pubblica dell'altra, o perché l'una ha prelevato dalla directory il certificato dell'altra o perché il certificato è contenuto nel messaggio iniziale che le parti coinvolte si sono scambiate.

La Figura 4.6 illustra le tre procedure.

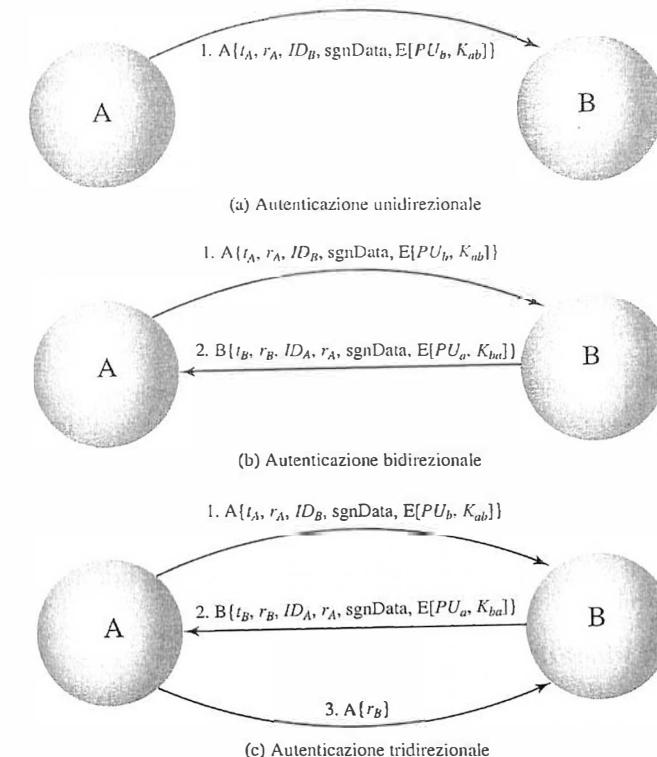


Figura 4.6 Procedure di autenticazione forte di X.509.

Autenticazione unidirezionale

L'autenticazione unidirezionale si basa su un unico trasferimento di informazioni da un utente A a un utente B, e stabilisce:

1. l'identità di A e il fatto che il messaggio sia stato generato da A;
2. il fatto che il messaggio sia destinato a B;
3. l'integrità e l'originalità (cioè il fatto che non sia stato inviato più volte) del messaggio.

Si noti che, durante questo processo, viene verificata solamente l'identità dell'entità che dà inizio al processo e non quella dell'entità che risponde.

Come requisito minimo, il messaggio contiene un timestamp t_A , un nonce r_A e l'identità di B ed è firmato con la chiave privata di A. Il timestamp contiene un tempo di creazione opzionale e il tempo di fine validità. Questo impedisce la consegna ritardata dei messaggi. Il nonce può essere utilizzato per scoprire attacchi di-replay. Il valore del nonce deve essere

unico all'interno del tempo di validità del messaggio. B può quindi memorizzare il nonce fino alla sua scadenza e rifiutare ogni ulteriore messaggio con lo stesso nonce.

Per scopi di mera autenticazione, il messaggio è semplicemente utilizzato per presentare le credenziali a B. Il messaggio può però anche includere delle informazioni da trasferire. Queste, sgnData , sono anch'esse firmate, garantendone così autenticità e integrità. Il messaggio può anche essere utilizzato per trasferire una chiave di sessione a B, cifrata con la chiave pubblica di B.

Autenticazione bidirezionale

In aggiunta ai tre elementi precedentemente citati, l'autenticazione bidirezionale stabilisce anche:

4. l'identità di B e il fatto che il messaggio di risposta sia stato generato da B;
5. il fatto che il messaggio sia destinato ad A;
6. l'integrità e l'originalità della risposta.

L'autenticazione bidirezionale consente quindi ad entrambe le parti in comunicazione di verificare reciprocamente la propria identità.

Il messaggio di risposta contiene un nonce generato da A, utilizzato per validare la risposta. Tale messaggio contiene inoltre un timestamp e un nonce generato da B. Come nel caso precedente, il messaggio può includere ulteriori informazioni firmate, e una chiave di sessione cifrata con la chiave pubblica di A.

Autenticazione tridirezionale

Nell'autenticazione tridirezionale, viene aggiunto un messaggio finale da A a B, che contiene una copia firmata del nonce r_B . Lo scopo dell'autenticazione tridirezionale è quello di evitare la verifica dei timestamp: dal momento che entrambi i nonce sono rispediti all'altra parte, ciascuna delle parti può verificare il nonce che gli è stato restituito al fine di scoprire eventuali attacchi di replay. Questo approccio è necessario se non si hanno a disposizione orologi sincronizzati.

Versione 3 di X.509

Il formato della versione 2 di X.509 non soddisfa tutti i requisiti di cui la più recente esperienza, maturata attraverso la progettazione e l'implementazione, ha evidenziato la necessità. In [FORD95] sono elencati i requisiti non soddisfatti dalla versione 2.

1. Il campo soggetto non è adeguato a comunicare l'identità del possessore di una chiave a un utilizzatore di chiave pubblica. I nomi X.509 possono essere relativamente brevi e mancare quindi di ovvi dettagli di identificazione che possono essere necessari per l'utente.
2. Il campo soggetto è inoltre inadeguato per molte applicazioni che tipicamente riconoscono un'entità tramite un indirizzo di posta elettronica, un URL o altri meccanismi utilizzati in Internet per l'identificazione.
3. Esiste la necessità di fornire informazioni circa la politica di sicurezza. Questo consente a un'applicazione o funzione di sicurezza, come ad esempio IPSec, di mettere in relazione un certificato X.509 con una determinata politica.

4. Esiste la necessità di porre un limite ai danni che possono essere provocati da una CA malintenzionata o mal funzionante ponendo dei vincoli all'applicabilità di un dato certificato.
5. È importante essere in grado di identificare separatamente chiavi diverse utilizzate in tempi diversi dallo stesso proprietario. Questa funzionalità rende possibile la gestione del ciclo di vita delle chiavi, in particolar modo la capacità di aggiornare le coppie di chiavi per gli utenti e per le CA, regolarmente o nel caso di circostanze eccezionali.

Invece di continuare ad aggiungere campi a un formato fisso, gli svilupparori degli standard hanno ritenuto che fosse necessario sviluppare un approccio più flessibile. Per questo, la versione 3 contiene una serie di estensioni opzionali che possono essere aggiunte al formato previsto dalla versione 2. Ogni estensione consta di: un identificatore, un indicatore di criticità e un valore. L'indicatore di criticità serve per segnalare se l'estensione può essere ignorata senza arrecare problemi alla sicurezza. Se l'indicatore ha valore TRUE e un'implementazione non riconosce l'estensione, allora questa deve considerare non valido il certificato.

Le estensioni presenti in un certificato possono essere classificate in tre categorie principali: informazioni sulle chiavi e sulle politiche, attributi relativi al soggetto e a chi emette il certificato, e vincoli sul percorso di certificazione.

Informazioni sulle chiavi e sulle politiche

Queste estensioni forniscono informazioni addizionali circa le chiavi del soggetto e dell'entità che ha emanato il certificato. Inoltre, forniscono degli indicatori relativi alla politica dei certificati. Una politica riguardante i certificati identifica un insieme di regole cui è attribuito un nome che fornisce indicazioni sull'applicabilità di un certificato in una certa organizzazione e/o in un particolare contesto applicativo caratterizzato da requisiti di sicurezza comuni. Quest'area comprende:

- **Identificatore della chiave dell'autorità.** Identifica la chiave pubblica da utilizzare per verificare la firma sul certificato o sulla CRL. Consente di distinguere chiavi diverse appartenenti alla stessa CA. Uno scopo di questo campo è la gestione dell'aggiornamento della coppia di chiavi di una CA.
- **Identificatore della chiave del soggetto.** Identifica la chiave pubblica da certificare. È utile per l'aggiornamento delle coppie di chiavi dei soggetti. Inoltre, un soggetto può avere più coppie di chiavi e, corrispondentemente, diversi certificati per scopi diversi (per esempio, firma digitale e accordo sulla chiave di cifratura).
- **Utilizzo della chiave.** Pone restrizioni agli ambiti di utilizzo della chiave pubblica certificata e alle politiche con cui la chiave può essere utilizzata. Può indicare uno dei seguenti ambiti: firma digitale, non-ripudio, cifratura della chiave, cifratura dati, accordo sulla chiave, verifica della firma della CA sui certificati, verifica della firma della CA sulle CRL.
- **Periodo di utilizzo della chiave privata.** Indica il periodo di utilizzo della chiave privata corrispondente alla chiave pubblica. I periodi di validità della chiave pubblica e della chiave privata sono generalmente diversi. Ad esempio, nel caso di chiavi per la firma digitale, il periodo di utilizzo della chiave privata usata per la firma è solitamente più breve di quello della corrispondente chiave pubblica usata per verificare la firma.

- Politiche riguardanti i certificati.** I certificati possono essere utilizzati in contesti in cui si applicano molteplici politiche. Questa estensione elenca le politiche previste dal certificato, insieme a ulteriori informazioni opzionali.
- Corrispondenze tra politiche.** Questa estensione è utilizzata solo nel caso di certificati per CA emanati da altre CA. Le corrispondenze tra politiche permettono alla CA che emette il certificato di indicare che una o più politiche locali alla CA possono essere considerate equivalenti a un'altra politica utilizzata nel dominio della CA cui il soggetto appartiene.

Attributi del certificato relativi al soggetto e a chi lo emette

Queste estensioni consentono di specificare nomi alternativi, con formati alternativi, sia per il soggetto cui il certificato si riferisce sia per l'entità che ha emesso il certificato. Le informazioni aggiuntive che tali estensioni forniscono circa i soggetti possono essere utilizzate per aumentare il livello di fiducia nel fatto che il soggetto cui il certificato corrisponde sia una certa persona o entità. Ad esempio, possono essere necessarie informazioni come l'indirizzo di posta, la posizione ricoperta all'interno di un'organizzazione o una foto.

I campi delle estensioni che fanno parte di questo gruppo includono:

- Nome alternativo del soggetto.** Contiene uno o più nomi alternativi, in un formato qualsiasi, scelto tra un insieme di formati. Questo campo è importante per determinati tipi di applicazioni – quali posta elettronica, EDI e IPSec – che possono utilizzare formati proprietari per i nomi.
- Nome alternativo dell'entità che ha emanato il certificato.** Contiene uno o più nomi alternativi, in un formato qualsiasi, scelto tra un insieme di formati.
- Attributi relativi alla directory del soggetto.** Fornisce il valore di un qualsiasi attributo della directory X.500 per il soggetto cui si riferisce il certificato.

Vincoli sul percorso di certificazione

Queste estensioni consentono di specificare i vincoli da includere all'interno di certificati emanati da CA per altre CA. I vincoli possono imporre restrizioni sul tipo di certificati che possono essere emessi dalla CA cui il soggetto fa riferimento o dalle CA che possono essere contenute in un percorso di certificazione.

I campi delle estensioni che fanno parte di questo gruppo sono i seguenti.

- Vincoli di base.** Campo che indica se il soggetto può ricoprire il ruolo di CA. In questo caso, può essere specificato un vincolo sulla lunghezza del percorso di certificazione.
- Vincoli sul nome.** Campo che specifica uno spazio dei nomi in cui devono essere contenuti tutti i nomi dei soggetti che appaiono nei successivi certificati nel percorso di certificazione.
- Vincoli sulla politica.** Campo che specifica vincoli che possono richiedere l'identificazione esplicita di una politica riguardante i certificati o non consentire di stabilire corrispondenze tra politiche per la restante parte del percorso di certificazione.

4.3 Infrastruttura per la chiave pubblica

In RFC 2822 (*glossario della sicurezza Internet*) viene definita l'infrastruttura per la chiave pubblica (PKI, *public key infrastructure*) come un insieme di hardware, software, persone, politiche e procedure necessarie per creare, gestire, memorizzare, distribuire e revocare certificati digitali basati sulla crittografia asimmetrica. Il principale obiettivo nello sviluppo di una PKI è quello di abilitare un'acquisizione sicura, comoda ed efficiente delle chiavi pubbliche. Il gruppo di lavoro di Internet Engineering Task Force (IETF) sull'infrastruttura per la chiave pubblica X.509 (PKIX, *public key infrastructure X.509*) è stato la forza trainante per la creazione di un modello formale (e generico) basato su X.509, adatto per realizzare un'architettura basata sui certificati, su Internet.

Nella Figura 4.7 sono illustrate le interrelazioni tra gli elementi chiave del modello PKIX. Questi elementi sono:

- Entità finale:** termine generico per indicare gli utenti finali, i dispositivi (ad esempio, server e router) o una qualsiasi altra entità che può essere identificata nel campo del soggetto di un certificato a chiave pubblica. Le entità finali tipicamente utilizzano e/o supportano i servizi legati alla PKI.
- Autorità di certificazione (CA, certification authority):** l'elemento che emette i certificati e (di solito) le liste di revoca dei certificati (CRL, *certificate revocation list*). Potrebbe anche supportare diverse funzioni amministrative, sebbene queste siano spesso delegate a una o più autorità di registrazione.

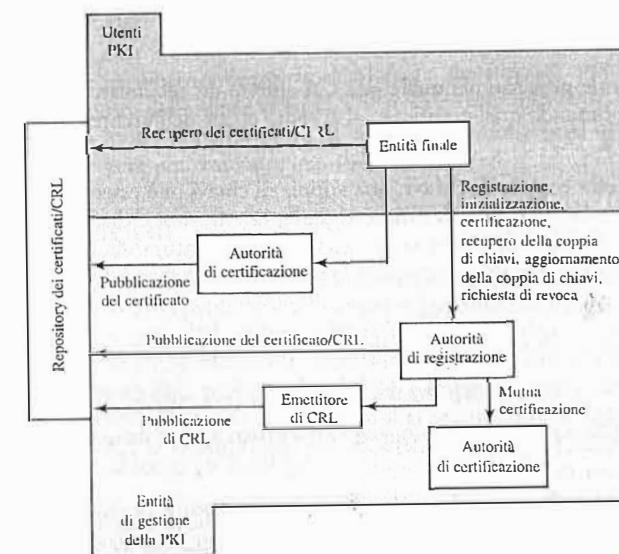


Figura 4.7 Modello Architetturale di PKIX.

- **Autorità di registrazione** (RA, *registration authority*): componente opzionale che può svolgere parecchie funzioni amministrative della CA. La RA è sovente associata al processo di registrazione dell'entità finale, ma può essere d'aiuto anche in molte altre aree.
- **Emettitore di CRL**: componente opzionale che la CA può delegare per pubblicare le CRL.
- **Repository**: termine generico usato per indicare un qualsiasi metodo per memorizzare certificati e CRL in modo che possano essere recuperati dalle entità finali.

Funzioni di gestione della PKIX

PKIX identifica parecchie funzioni che potenzialmente dovrebbero essere supportate dai protocolli di gestione supportassero. Queste sono indicate nella Figura 4.7 e comprendono:

- **Registrazione**: processo per mezzo del quale, in primo luogo, un utente si fa conoscere alla CA (direttamente o attraverso una RA) prima che questa emetta uno o più certificati per quell'utente. La registrazione inizia con il processo di iscrizione a una PKI e di solito è costituita da alcune procedure off-line o on-line per la mutua autenticazione. Tipicamente, all'entità finale vengono rilasciate una o più chiavi segrete condive per la successiva autenticazione.
- **Inizializzazione**: prima che un sistema client possa funzionare in modo sicuro, è necessario installare l'occorrente per le chiavi che ha una relazione appropriata con le chiavi memorizzate in qualche altro punto dell'infrastruttura. Ad esempio, il client ha la necessità di essere inizializzato in modo sicuro con la chiave pubblica e altre informazioni sicure della o delle CA fidate, per essere usate nella validazione dei cammini dei certificati.
- **Certificazione**: processo nel quale una CA emette un certificato a chiave pubblica di un utente e rimanda quel certificato al sistema client dell'utente e/o lo memorizza in un repository.
- **Recupero della coppia di chiavi**: una coppia di chiavi può essere usata per consentire la creazione e la verifica della firma digitale, la cifratura e decifratura, o entrambe. Quando una coppia di chiavi viene usata per la cifratura/decifratura è importante fornire un meccanismo per recuperare le indispensabili chiavi di decifratura – quando non è più possibile un normale accesso all'occorrente per le chiavi –, altrimenti non sarà possibile recuperare i dati cifrati. La perdita dell'accesso alle chiavi di decifratura può essere conseguenza di password/PIN dimenticati, di dischi rigidi rovinati, di token hardware danneggiati e così via. Il recupero della coppia di chiavi consente alle entità finali di ripristinare la loro coppia di chiavi di cifratura/decifratura da un sistema di backup autorizzato della chiave (tipicamente la CA che ha emesso il certificato dell'entità finale).
- **Aggiornamento della coppia di chiavi**: tutte le coppie di chiavi hanno la necessità di essere aggiornate regolarmente (cioè di essere rimpiazzate da una nuova coppia di chiavi) e vanno emessi nuovi certificati. L'aggiornamento è richiesto quando il tempo di vita del certificato scade o come conseguenza di una revoca del certificato.

- **Richiesta di revoca**: una persona autorizzata avvisa una CA di una situazione anomala, richiedendo la revoca del certificato. Le ragioni della revoca comprendono la compromissione delle chiavi private, una modifica nell'affiliazione e un cambiamento del nome.
- **Mutua certificazione**: due CA si scambiano informazioni che vengono usate per stabilire una mutua certificazione. Un mutuo certificato è un certificato emesso da una CA a un'altra CA e che contiene la chiave di firma della CA, usata per emettere i certificati.

Protocolli di gestione PKIX

Il gruppo di lavoro PKIX ha definito due protocolli di gestione alternativi tra le entità PKIX, che supportano le funzioni di gestione elencate nel precedente sotto paragrafo. Nella RFC 2510 è definito il protocollo di gestione dei certificati (CMP, *certificate management protocol*), all'interno del quale ciascuna delle funzioni di gestione è esplicitamente identificata da specifici scambi del protocollo. CMP è progettato per essere un protocollo flessibile, capace di adattarsi a diversi modelli tecnici, operazionali e di business.

Nella RFC 2797 sono definiti i messaggi di gestione dei certificati su CMS (CMC, *certificate management over CMS*) dove per CMS si fa riferimento alla RFC 2630 e si intende la sintassi dei messaggi crittografici (CMS, *cryptographic message syntax*). CMS è realizzato basandosi su un lavoro precedente e si propone di far leva sulle implementazioni esistenti. Sebbene tutte le funzioni PKIX siano supportate, non si traducono in specifici scambi del protocollo.

4.4 Letture consigliate e siti web

La lettura di [BRYA88] costituisce un modo semplice per avere un'idea dei concetti alla base di Kerberos. Una delle migliori trattazioni su Kerberos è [KOHL94]. [TUNG99] descrive Kerberos dal punto di vista dell'utente.

[PERL99] rivisita molti modelli di fiducia che possono essere usati in una PKI e consiglia gli approcci da seguire per realizzare una PKI efficace.

Siti web consigliati

- **Sito di Kerberos presso il MIT**: contiene informazioni su Kerberos, incluse le FAQ, articoli e documenti sull'argomento e riferimenti a siti web relativi a prodotti commerciali.
- **Pagina di Kerberos USC/ISI**: è un'altra valida sorgente di informazioni su Kerberos.
- **Kerberos Working Group**: gruppo di IETF che sviluppa standard basati su Kerberos.
- **Gruppo di lavoro sull'infrastruttura a chiave pubblica**: gruppo IETF preposto allo sviluppo di standard basati su X.509v3.
- **VeriSign**: è uno dei fornitori commerciali leader nel settore dei prodotti connessi a X.509; nel sito VeriSign è possibile trovare articoli e altro materiale di interesse sull'argomento.
- **NIST PKI Program**: buona fonte di informazioni.

4.5 Domande di revisione ed esercizi

Domande di revisione

- 4.1 Per rispondere a quali problemi è stato sviluppato Kerberos?
- 4.2 Quali sono le tre minacce associate all'autenticazione dell'utente su una rete o su Internet?
- 4.3 Elencate i tre approcci per rendere sicura l'autenticazione dell'utente in un ambiente distribuito.
- 4.4 Quali sono i quattro requisiti definiti per Kerberos?
- 4.5 Quali entità costituiscono un ambiente Kerberos comprensivo (*full-service*)?
- 4.6 Nel contesto di Kerberos, che cos'è un realm?
- 4.7 Quali sono le principali differenze tra la versione 4 e 5 di Kerberos?
- 4.8 Qual è lo scopo dello standard X.509?
- 4.9 Che cos'è una catena di certificati?
- 4.10 Come viene revocato un certificato X.509?

Esercizi

- 4.1 Dimostrate che in modalità PCBC un errore casuale in un blocco di testo cifrato si propaga a tutti i successivi blocchi di testo in chiaro (Figura 4.9).
- 4.2 Si supponga che, in modalità PCBC, i blocchi C_i e C_{i+1} siano scambiati durante la trasmissione. Mostrate come questo scambio influenzi solo i blocchi decifrati P_i e P_{i+1} e non i blocchi successivi.
- 4.3 La formulazione iniziale della procedura di autenticazione tridirezionale di X.509, illustrata nella Figura 4.6c, contiene un punto debole per quanto riguarda la sicurezza. L'essenza del protocollo è la seguente:

$$\begin{aligned} A \rightarrow B: & A \{t_A, r_A, ID_B\} \\ B \rightarrow A: & B \{t_B, r_B, ID_A, r_A\} \\ A \rightarrow B: & A \{r_B\} \end{aligned}$$

X.509 stabilisce che la verifica dei timestamp t_A e t_B è opzionale nel caso di autenticazione tridirezionale. A tal proposito, considerate il seguente esempio. Supponete che A e B abbiano utilizzato il protocollo sopra descritto in una precedente occasione, e che l'avversario C abbia intercettato i tre messaggi. Inoltre, assumete che i timestamp non siano utilizzati e siano posti tutti a 0. Infine, si supponete che C voglia impersonare A nei confronti di B. C inizia a spedire il primo messaggio intercettato:

$$C \rightarrow B: A \{0, r_A, ID_B\}$$

B risponde, pensando di aver a che fare con A, mentre in realtà sta comunicando con C:

$$B \rightarrow C: B \{0, r'_B, ID_A, r_A\}$$

Nel frattempo C, utilizzando un qualche mezzo, fa in modo che A inizi l'autenticazione con C. Il risultato è che A manda a C il seguente messaggio:

$$A \rightarrow C: A \{0, r'_A, ID_C\}$$

C risponde ad A, utilizzando lo stesso nonce che B ha fornito a C.

$$C \rightarrow A: C \{0, r'_B, ID_A, r'_A\}$$

A risponde con il messaggio

$$A \rightarrow C: A \{r'_B\}$$

Tale messaggio è esattamente quello che serviva a C per convincere B del fatto che sta comunicando con A; a questo punto C rispedisce il messaggio a B.

$$C \rightarrow B: A \{r'_B\}$$

B crederà di comunicare con A, invece sta comunicando con C. Determinate una semplice soluzione a questo problema che non richieda l'utilizzo di timestamp.

- 4.4 La versione 1988 di X.509 elenca le proprietà che le chiavi RSA devono soddisfare per essere sicure, sulla base della difficoltà che tutt'oggi sussistono nel fattorizzare numeri elevati. La discussione si conclude con un vincolo sull'esponente e sul modulo n :

Deve essere assicurato il soddisfacimento della seguente diseguaglianza $e > \log_2(n)$ per prevenire attacchi che scoprano il testo in chiaro utilizzando la e -esima radice mod n .

Sebbene tale vincolo sia corretto, non è corretta la motivazione per cui si richiede il suo soddisfacimento. Cosa c'è di sbagliato nella motivazione data e qual è quella corretta?

Appendice 4A Tecniche di cifratura di Kerberos

Kerberos contiene una libreria a supporto della cifratura e delle operazioni ad essa correlate. Queste furono incluse nella specifica di Kerberos 5 e sono comuni nelle implementazioni commerciali. Nel febbraio del 2005, IETF emise le RFC 3961 e 3962, che ampliavano le opzioni delle tecniche critografiche. In questo appendice vengono descritte le tecniche della RFC 1510.

Trasformazione da password a chiave

In Kerberos le password possono contenere solo i caratteri rappresentabili tramite il formato ASCII a 7 bit. La password, che può avere una lunghezza arbitraria, è trasformata in una chiave di ciphatura memorizzata nella base di dati di Kerberos. La Figura 4.8 illustra tale trasformazione.

In primo luogo, la stringa di caratteri s è codificata in una stringa di bit b, in modo che il primo carattere sia memorizzato nei primi 7 bit, il secondo nei secondi 7 bit, e così via. Questa codifica può essere rappresentata come segue:

$$\begin{aligned} b[0] &= \text{bit 0 di } s[0] \\ \vdots & \\ b[6] &= \text{bit 6 di } s[0] \\ b[7] &= \text{bit 0 di } s[1] \\ \vdots & \\ b[7i + m] &= \text{bit } m \text{ di } s[i] \quad 0 \leq m \leq 6 \end{aligned}$$

Successivamente la stringa di bit è compattata in 56 bit tramite un allineamento dei bit ed effettuando opportune operazioni di OR esclusivo bit a bit. Ad esempio, se la stringa di bit ha lunghezza 59, allora:

$$\begin{aligned} b[55] &= b[55] \oplus b[56] \\ b[54] &= b[54] \oplus b[57] \\ b[53] &= b[53] \oplus b[58] \end{aligned}$$

In questo modo si genera una chiave DES a 56 bit. Per conformarsi al formato di chiave richiesto a 64 bit, la stringa è considerata come una sequenza di otto blocchi di 7 bit ed è convertita in otto blocchi di 8 bit per ottenere la chiave di ingresso K_{pw} .

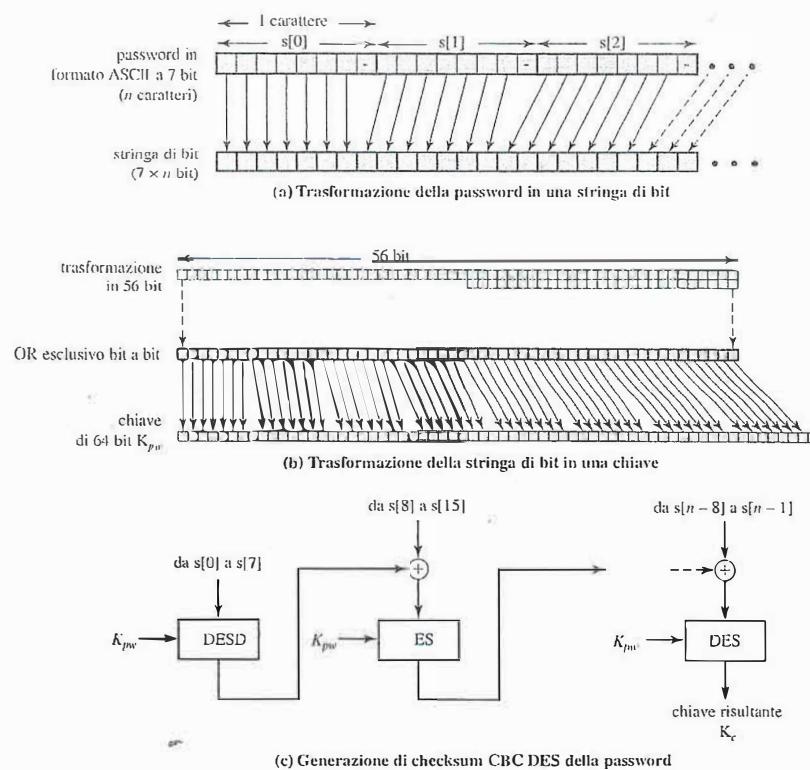


Figura 4.8 Generazione della chiave di ciphertura dalla password.

Infine, la password originale è ciphertata utilizzando la modalità *cipher block chaining* (CBC) di DES mediante la chiave K_{pw} . L'ultimo blocco di 64 bit restituito da questo processo, conosciuto come checksum CBC, costituisce la chiave risultante associata alla password.

L'algoritmo nel suo complesso può essere visto come una funzione hash che trasforma una password arbitraria in un codice hash a 64 bit.

Modalità PCBC

Si ricordi, dal Capitolo 2, che nella modalità CBC di DES, l'ingresso dell'algoritmo a ogni iterazione consiste nell'applicazione dell'operazione di OR esclusivo al blocco di testo in chiaro corrente ed al precedente blocco di testo cipherto, utilizzando la stessa chiave per ciascun blocco (Figura 2.9). Il vantaggio di questa modalità rispetto alla modalità *electronic codebook* (ECB), in cui ogni blocco di testo in chiaro è ciphertato in modo indipendente, è che in modalità CBC lo stesso blocco di testo in chiaro, se ripetuto, dà origine a differenti blocchi di testo cipherto.

La modalità CBC è strutturata in modo che se si verifica un errore nella trasmissione del blocco cipherto C_t , questo errore si propaga ai blocchi di testo in chiaro P_t e P_{t+1} .

La versione 4 di Kerberos utilizza un'estensione di CBC, chiamata modalità propagating CBC (PCBC, *propagating cipher block chaining*) [MEYE82]. Questa modalità ha la proprietà che un errore in un blocco di testo cipherto viene propagato a tutti i successivi blocchi decifrati presenti nel messaggio rendendoli, quindi, inutilizzabili. La ciphertura dati e l'integrità sono così combinate in un'unica operazione. Per una eccezione si veda il Problema 4.2.

La modalità PCBC è illustrata nella Figura 4.9. In questo schema, l'ingresso dell'algoritmo di ciphertura è l'OR esclusivo del corrente blocco di testo in chiaro, del precedente blocco di testo cipherto, e del precedente blocco di testo in chiaro:

$$C_n = E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])$$

Durante la decifrazione, ogni blocco di testo cipherto è elaborato dall'algoritmo di decifrazione. L'uscita di questo algoritmo è composta mediante la funzione di OR esclusivo con il precedente blocco di testo cipherto e il precedente blocco di testo in chiaro. Si può dimostrare che questo schema funziona, nel seguente modo:

$$D(K, C_n) = D(K, E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n]))$$

$$D(K, C_n) = C_{n-1} \oplus P_{n-1} \oplus P_n$$

$$C_{n-1} \oplus P_{n-1} \oplus D(K, C_n) = P_n$$

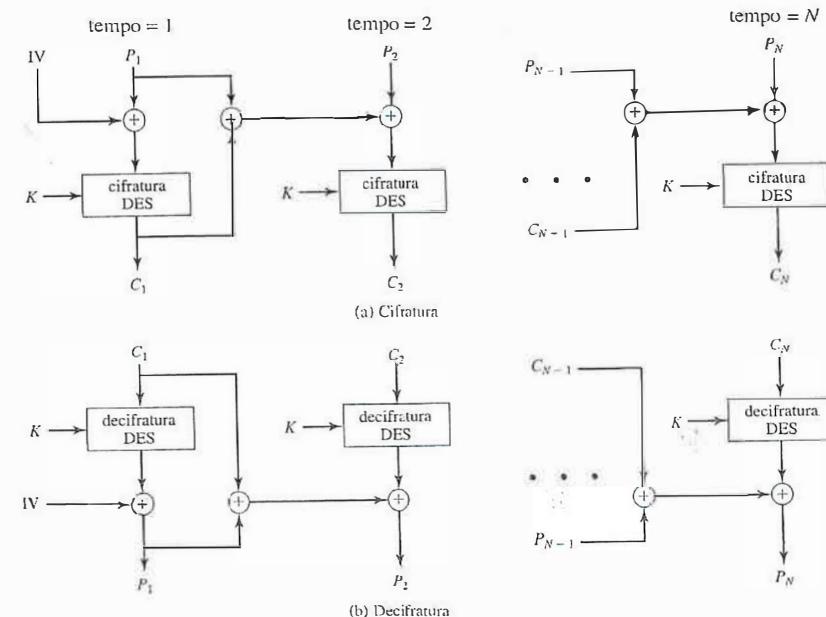


Figura 4.9 Modalità PCBC.

Capitolo 5

Sicurezza della posta elettronica

In tutti gli ambienti distribuiti, la posta elettronica è l'applicazione di rete maggiormente utilizzata ed è l'unica applicazione distribuita largamente impiegata su architetture e piattaforme commerciali diverse. Gli utenti possono inviare posta ad altri utenti connessi direttamente o indirettamente a Internet, senza preoccuparsi del loro sistema operativo o degli applicativi di comunicazione di cui dispongono.

Con la crescente tendenza a sfruttare la posta elettronica per gli scopi più diversificati, aumenta la conseguente richiesta di servizi di autenticazione e riservatezza. Due sono i principali approcci candidati a un'ampia diffusione nei prossimi anni: PGP (*pretty good privacy*) e S/MIME (*secure/multipurpose internet mail extensions*), entrambi esaminati in questo capitolo.

5.1 Pretty good privacy

PGP costituisce un fenomeno degno di nota. Essenzialmente legato al lavoro di ricerca di Phil Zimmermann, PGP fornisce un servizio di autenticazione e riservatezza che può essere utilizzato per applicazioni di posta elettronica e di archiviazione su file. In sostanza, questo è il percorso seguito da Zimmermann.

1. Inizialmente, selezionò come elementi costitutivi i migliori algoritmi di crittografia disponibili.
2. In seguito, integrò questi algoritmi in un'applicazione di uso generale, indipendente dal sistema operativo e dal processore, basata su un insieme ridotto di semplici comandi.
3. Rese quindi il pacchetto e la sua documentazione, codice sorgente incluso, disponibili gratuitamente su Internet, su BBS (*Bulletin Boards*) e su reti commerciali quali, ad esempio, AOL (*America on line*).
4. Stipulò infine un accordo con una società (Viacrypt, oggi Network Associates) per produrre, a costi contenuti, una versione commerciale di PGP completamente compatibile.

L'ampia diffusione e l'esteso utilizzo di cui gode attualmente PGP sono legati a una serie di ragioni.

- È disponibile gratuitamente in versioni che girano su molteplici piattaforme fra cui Windows, UNIX, Macintosh e molte altre. Inoltre, la versione commerciale soddisfa le esigenze degli utenti che desiderano un prodotto con assistenza da parte del fornitore.
- PGP è basato su algoritmi che hanno superato approfondite revisioni pubbliche e che sono quindi considerati estremamente sicuri. In particolare, il pacchetto comprende gli algoritmi RSA, DSS e Diffie-Hellman per la cifratura a chiave pubblica; gli algoritmi CAST-128, IDEA e 3DES per la cifratura convenzionale, e l'algoritmo SHA-1 per la codifica hash.
- L'applicabilità di PGP è ampia: dalle imprese che desiderano adottare uno schema standardizzato per la cifratura di file e messaggi ai singoli individui desiderosi di comunicare in maniera sicura attraverso Internet e altre reti di comunicazione con altri utenti localizzati ovunque.
- PGP non è stato sviluppato né è controllato da organizzazioni governative o di standardizzazione. Questo lo rende particolarmente interessante per quanti hanno un'avversione istintiva nei confronti dell'*establishment*.
- PGP è ora uno standard Internet (RFC 3156), tuttavia ha ancora un'aura di attività anti-istituzionale.

La trattazione di PGP incomincia con la descrizione operativa. Successivamente, saranno considerate la creazione e la memorizzazione delle chiavi crittografiche e verrà affrontato il problema essenziale della gestione delle chiavi pubbliche.

Notazione

Gran parte delle notazioni usate in questo capitolo sono già state utilizzate, ma alcuni elementi sono nuovi. Ecco una sintesi della notazione, che si basa sui seguenti simboli.

K_s = chiave di sessione usata negli schemi di cifratura convenzionale

PR_a = chiave privata dell'utente A, usata negli schemi di cifratura a chiave pubblica

PU_a = chiave pubblica dell'utente A, usata negli schemi di cifratura a chiave pubblica

EP = cifratura a chiave pubblica

DP = decifratura a chiave pubblica

EC = cifratura simmetrica

DC = decifratura simmetrica

H = funzione hash

\parallel = concatenazione

Z = compressione con l'algoritmo ZIP

R64 = conversione al formato ASCII radix-64

La documentazione PGP fa spesso uso del termine **chiave segreta** per indicare una chiave utilizzata in coppia con una chiave pubblica in uno schema di cifratura a chiave pubblica.

Questo può generare confusione con la chiave segreta usata nell'ambito della cifratura simmetrica. Pertanto, si utilizzerà al suo posto il termine **chiave privata**.

Descrizione operativa

Il funzionamento attuale di PGP, al contrario della gestione delle chiavi, prevede cinque servizi: autenticazione, riservatezza, compressione, compatibilità con la posta elettronica e segmentazione (Tabella 5.1).

Autenticazione

La Figura 5.1a mostra il servizio di firma digitale fornito da PGP. Questo schema coincide con quello discusso nel Capitolo 3 e illustrato nella Figura 3.2b. Si ha la sequenza di seguito riportata.

- Il mittente crea un messaggio.
- Si utilizza SHA-1 per generare un codice hash di 160 bit del messaggio.

Funzionalità	Algoritmi utilizzati	Descrizione
Firma digitale	DSS/SHA o RSA/SHA	Si genera un codice hash del messaggio con SHA-1. Questo digest di messaggio viene cifrato con DSS o RSA, utilizzando la chiave privata del mittente, e viene quindi incluso nel messaggio.
Cifratura del messaggio	CAST o IDEA o triplo DES	Il messaggio viene cifrato con CAST-128 o a tre chiavi con Diffie-Hellman IDEA o triplo DES con una chiave di RSA sessione one-time creata dal mittente. Questa chiave di sessione viene cifrata con la chiave pubblica del ricevente usando l'algoritmo Diffie-Hellman o RSA e viene inclusa nel messaggio.
Compressione	ZIP	Il messaggio può essere compresso utilizzando ZIP per scopi di memorizzazione o trasmissione.
Compatibilità con la posta elettronica	Conversione radix-64	Un messaggio cifrato può essere convertito in una stringa ASCII utilizzando la conversione radix-64, al fine di fornire trasparenza per le applicazioni di posta elettronica.
Segmentazione	—	Per adattarsi alle limitazioni sull'lunghezza massima dei messaggi, PGP esegue la segmentazione e il riassembaggio di un messaggio.

Tabella 5.1 Prospetto riassuntivo dei servizi di PGP.

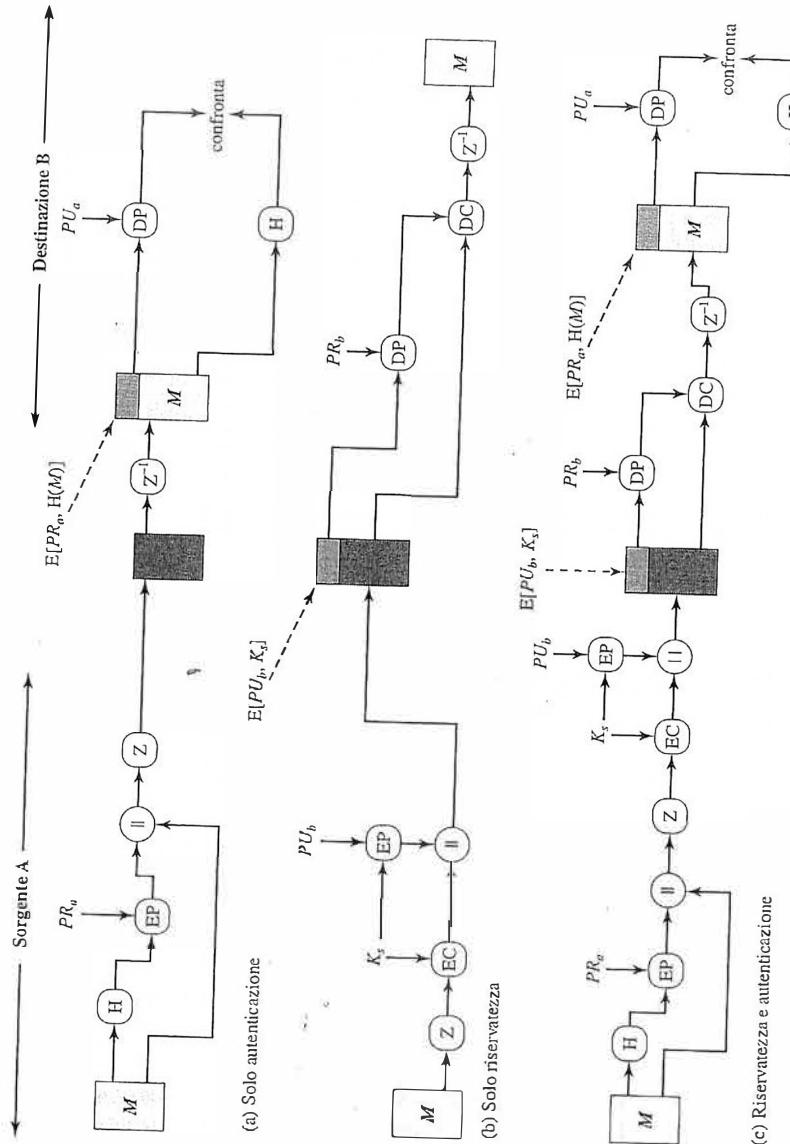


Figura 5.1 Funzionalità crittografiche di PGP

3. Il codice hash viene cifrato con RSA usando la chiave privata del mittente e il risultato così ottenuto viene apposto al messaggio.
4. Il ricevente usa RSA con la chiave pubblica del mittente per decifrare ciò che ha ricevuto e ricostruire il codice hash.
5. Il ricevente genera un nuovo codice hash per il messaggio e lo confronta con quello ottenuto dal processo di decifratura. Se i due codici coincidono, il messaggio viene accettato come autentico.

L'uso combinato di SHA-1 e RSA fornisce uno schema di firma digitale efficace. Data la robustezza di RSA, il ricevente ha garanzia che soltanto il possessore della corrispondente chiave privata è in grado di generare la firma. Data la robustezza di SHA-1, il ricevente ha garanzia che nessun altro potrebbe generare un nuovo messaggio che abbia lo stesso codice hash e, quindi, la stessa firma di quello originario. In alternativa, si potrebbero generare le firme utilizzando la combinazione DSS/SHA-1.

Anche se normalmente le firme sono apposte al messaggio o al file oggetto di firma, talvolta sono supportate anche firme separate (*detached*). Una firma separata può essere memorizzata e trasmessa separatamente dal messaggio oggetto di firma. Questa modalità può essere utile in molti contesti. Un utente potrebbe voler mantenere un file di log delle firme di tutti i messaggi inviati o ricevuti. Una firma separata di un programma eseguibile può rivelare in tempi successivi un'infezione prodotta da un virus. Infine, le firme separate possono essere utilizzate quando un documento deve essere firmato da più di una parte come, ad esempio, in caso di contratti. La firma di ciascuna persona è indipendente dalle altre e viene quindi apposta solo al documento. In caso contrario, le firme dovrebbero essere poste in maniera nidificata, ovvero il secondo firmatario dovrebbe firmare sia il documento sia la firma del primo firmatario, e così via.

Riservatezza

Un altro servizio base fornito da PGP è la riservatezza, realizzata mediante la cifratura dei messaggi da trasmettere o da memorizzare localmente come file. In entrambi i casi, si può usare l'algoritmo di cifratura convenzionale CAST-128. In alternativa, possono essere usati anche gli algoritmi IDEA o 3DES. Viene usata la modalità *cipher feedback* (CFB) a 64 bit.

Come sempre, occorre affrontare il problema della distribuzione delle chiavi. In PGP, ciascuna chiave simmetrica viene usata una sola volta. In altre parole, si genera una nuova chiave sotto forma di numero casuale a 128 bit per ogni messaggio. In sostanza si tratta di una chiave *one-time* (usa e getta), anche se nella documentazione viene riferita come chiave di sessione. Dato che deve essere usata una sola volta, la chiave di sessione viene legata al messaggio e trasmessa con esso. Questa chiave viene protetta cifrandola con la chiave pubblica del ricevente. La Figura 5.1b mostra la sequenza di operazioni.

1. Il mittente crea un messaggio e un numero casuale di 128 bit da utilizzare come chiave di sessione per il solo messaggio in questione.
2. Il messaggio viene cifrato con la chiave di sessione utilizzando CAST-128 (o IDEA o 3DES).
3. La chiave di sessione viene cifrata utilizzando RSA, con la chiave pubblica del ricevente, e apposta al messaggio.

4. Il ricevente utilizza RSA con la propria chiave privata per decifrare e ricostruire la chiave di sessione.
5. La chiave di sessione viene utilizzata per decifrare il messaggio.

Come alternativa all'impiego di RSA per la cifratura della chiave, PGP fornisce un'opzione denominata Diffie-Hellman. Come descritto nel Capitolo 3, Diffie-Hellman è un algoritmo per lo scambio delle chiavi. PGP usa una variante dell'algoritmo Diffie-Hellman, conosciuta come **ElGamal**, che non fornisce cifratura/decifratura (Esercizio 10.6).

Si possono fare una serie di considerazioni. Innanzitutto, per ridurre il tempo di cifratura si preferisce l'impiego combinato della cifratura convenzionale e di quella a chiave pubblica, rispetto al semplice impiego di RSA o ElGamal, per cifrare direttamente il messaggio: CAST-128 e gli altri algoritmi di cifratura convenzionale sono sostanzialmente più veloci di RSA o ElGamal. Secondariamente, l'impiego di un algoritmo a chiave pubblica risolve il problema della distribuzione della chiave di sessione, in quanto solo il ricevente è in grado di ricostruire la chiave di sessione inclusa nel messaggio. Si noti che non è necessario un protocollo di scambio della chiave di sessione, del tipo discusso nel Capitolo 3, perché non si sta iniziando una sessione continuativa. Piuttosto, ciascun messaggio e la sua chiave sono da considerarsi come un evento indipendente che accade una sola volta (*one-time*). Inoltre, dato il funzionamento della posta elettronica secondo la modalità "memorizza e inoltra", l'utilizzo dell'handshaking, per garantire che entrambe le parti abbiano la chiave di sessione, non è pratica. Infine, l'uso di chiavi convenzionali one-time rende ancora più robusto un approccio basato su cifratura simmetrica di per sé già robusto. Con ciascuna chiave viene cifrata solo una piccola porzione di testo in chiaro e non vi è relazione fra le diverse chiavi. Pertanto, a patto che l'algoritmo a chiave pubblica sia sicuro, l'intero schema risulta sicuro. A tal fine, PGP fornisce all'utente un intervallo di possibili lunghezze di chiave da 768 a 3072 bit (la chiave DSS per le firme è limitata a 1024 bit).

Riservatezza e autenticazione

Come mostrato nella Figura 5.1c, si possono usare entrambi i servizi di riservatezza e autenticazione per lo stesso messaggio. Prima di tutto si genera una firma per il messaggio in chiaro e la si appone al messaggio. Successivamente, il messaggio in chiaro e la firma vengono cifrati con CAST-128 (o con IDEA o 3DES), mentre la chiave di sessione viene cifrata con RSA (o ElGamal). Questa sequenza di operazioni è preferibile a quella inversa che prevederebbe in primo luogo la cifratura del messaggio e successivamente la generazione della firma per il messaggio cifrato. In generale è più conveniente memorizzare la firma con la versione in chiaro del messaggio. Inoltre, per scopi di autenticazione tramite una terza parte, eseguendo prima la firma, la terza parte può prescindere dalla chiave simmetrica nella verifica della firma.

In sintesi, quando vengono utilizzati entrambi i servizi, il mittente prima firma il messaggio con la propria chiave privata, poi cifra il messaggio con la chiave di sessione, e infine cifra la chiave di sessione con la chiave pubblica del ricevente.

Compressione

PGP esegue di default la compressione del messaggio, dopo aver applicato la firma e prima di eseguire la cifratura, con il vantaggio di risparmiare spazio di memoria sia per la trasmissione con la posta elettronica sia per la memorizzazione su file.

La collocazione dell'algoritmo di compressione, denotato nella Figura 5.1 con Z per la fase di compressione e con Z^{-1} per la fase di decompressione, è un aspetto critico.

1. La firma viene generata prima della compressione per due ragioni.
 - a. È preferibile firmare un messaggio non compresso in modo che, per successive verifiche, è possibile memorizzare solo il messaggio non compresso insieme alla sua firma. La firma dopo la compressione renderebbe necessaria, in alternativa, o la memorizzazione anche della versione compressa del messaggio, per le successive verifiche, oppure la ricompressione del messaggio in fase di verifica.
 - b. Anche volendo generare dinamicamente un messaggio ricompresso per scopi di verifica, l'algoritmo di compressione di PGP presenta una difficoltà. L'algoritmo è non deterministico: diverse implementazioni dell'algoritmo consentono di ottenere diversi compromessi fra velocità di esecuzione e percentuale di compressione, producendo come risultato compressioni differenti. Questi diversi algoritmi di compressione sono comunque interoperabili, nel senso che ogni versione dell'algoritmo può decomprimere correttamente il formato compresso ottenuto con una qualunque altra versione. L'applicazione della funzione hash e della firma dopo la fase di compressione vincolerebbe tutte le implementazioni PGP a una stessa versione dell'algoritmo di compressione.
2. La cifratura del messaggio è applicata dopo averne eseguito la compressione, per aumentare il livello di sicurezza offerto dalla crittografia. Dato che il messaggio compresso ha minore ridondanza del corrispondente messaggio in chiaro, la crittoanalisi diventa più difficile.

L'algoritmo di compressione utilizzato è ZIP, descritto nell'Appendice 5A.

Compatibilità con la posta elettronica

Utilizzando PGP, almeno una parte del blocco dati da trasmettere è cifrata. Se si utilizza solo il servizio di firma, viene cifrato il digest di messaggio (con la chiave privata del mittente). Se si utilizza il servizio di riservatezza, vengono cifrati il messaggio e (se presente) la firma (con una chiave simmetrica one-time). Pertanto, parte o tutto il blocco risultante è composto da un flusso di byte arbitrari. Tuttavia, molti sistemi di posta elettronica consentono di utilizzare solo blocchi composti da testo ASCII. Per adattarsi a tale restrizione, PGP fornisce un servizio di conversione del flusso binario grezzo a 8 bit in flusso di caratteri ASCII stampabili.

A tale scopo si impiega lo schema di conversione radix-64. Ogni gruppo di tre byte di dati binari viene convertito in quattro caratteri ASCII. Questo formato pone in coda al messaggio un CRC (*cyclic redundancy check*) per la rilevazione di errori di trasmissione. Una descrizione di questo processo è riportata nell'Appendice 5B.

L'uso dello schema radix-64 aumenta la dimensione di un messaggio del 33%. Fortunatamente, la chiave di sessione e la porzione di messaggio contenente la firma sono relativamente compatte, mentre il messaggio in chiaro è stato compresso. La compressione deve essere più che sufficiente per compensare l'espansione prodotta con radix-64. Ad esempio, [HELD96] riporta una percentuale media di compressione di circa 2.0 utilizzando ZIP. Trascurando le componenti relativamente piccole corrispondenti alle chiavi e alla firma, l'effetto globale tipico della compressione ed espansione di un file di lunghezza X sarebbe pari a $1,33 \times 0,5 \times X = 0,665 \times X$. Pertanto si ha in definitiva una compressione di circa un terzo.

Un aspetto degno di nota dell'algoritmo radix-64 riguarda il fatto che la conversione nel formato radix-64 viene eseguita indipendentemente dal flusso in ingresso, senza cioè considerare che il contenuto potrebbe anche essere già un testo a caratteri ASCII stampabili. Pertanto, se un messaggio viene firmato ma non cifrato e si effettua la conversione dell'intero blocco, il risultato sarebbe illeggibile a un osservatore casuale, il che fornisce un certo livello di riservatezza come opzione. PGP può essere configurato per eseguire la conversione nel formato radix-64 della sola porzione contenente la firma di un messaggio in chiaro firmato. Questo fa sì che l'utente che riceve il messaggio possa leggerlo senza utilizzare PGP che dovrebbe comunque essere utilizzato per verificare la firma.

La Figura 5.2 mostra la relazione fra i quattro servizi appena descritti. In fase di trasmissione, se richiesto, viene generata una firma utilizzando un codice hash del testo in chiaro non compresso. Quindi, se il messaggio è cifrato, il ricevente ricostruisce la chiave di sessione e decifra il messaggio. Il blocco risultante è poi decompresso. Se il messaggio è firmato, il ricevente ricostruisce il codice hash trasmesso e lo confronta con il codice hash risultante dai propri calcoli.

Al ricevimento del messaggio, il blocco dati in arrivo viene dapprima convertito dal formato radix-64 a quello binario. Quindi, se il messaggio è cifrato, il ricevente ricostruisce la chiave di sessione e decifra il messaggio. Il blocco risultante è poi decompresso. Se il messaggio è firmato, il ricevente ricostruisce il codice hash trasmesso e lo confronta con il codice hash risultante dai propri calcoli.

Segmentazione e riassemblaggio

Spesso gli strumenti di posta elettronica pongono restrizioni alla lunghezza massima dei messaggi. Ad esempio, alcuni strumenti accessibili via Internet impongono una lunghezza massima di 50.000 byte. Qualunque messaggio con lunghezza superiore deve essere suddiviso in segmenti più piccoli, ciascuno dei quali viene inviato separatamente.

Per adattarsi a tale restrizione, PGP suddivide automaticamente i messaggi troppo lunghi in segmenti di lunghezza inferiore, sufficiente per poter essere spediti attraverso posta elettronica. La segmentazione viene eseguita dopo tutte le altre elaborazioni, compresa la conversione radix-64. Pertanto, la componente contenente la chiave di sessione e quella contenente la firma appaiono una sola volta, all'inizio del primo segmento. Dal lato del ricevente, PGP deve eliminare tutte le intestazioni dei vari segmenti di posta elettronica e riassemblare il blocco originario nel suo insieme, prima di eseguire i passi mostrati nella Figura 5.2b.

Chiavi crittografiche e keyring

PGP fa uso di quattro tipi di chiavi: chiavi di sessione simmetriche one-time, chiavi pubbliche, chiavi private e chiavi simmetriche basate su *passphrase* (frase segreta usata come password), illustrate successivamente. Rispetto a queste chiavi, si possono identificare tre diversi requisiti.

1. È necessario uno strumento per la generazione di chiavi di sessione non prevedibili.
2. Un utente deve poter possedere molteplici copie chiave pubblica/chiave privata. Una ragione per tale requisito è che l'utente potrebbe voler cambiare la propria coppia di chiavi nel tempo. Quando questo accade, qualsiasi messaggio nella coda di ricezione

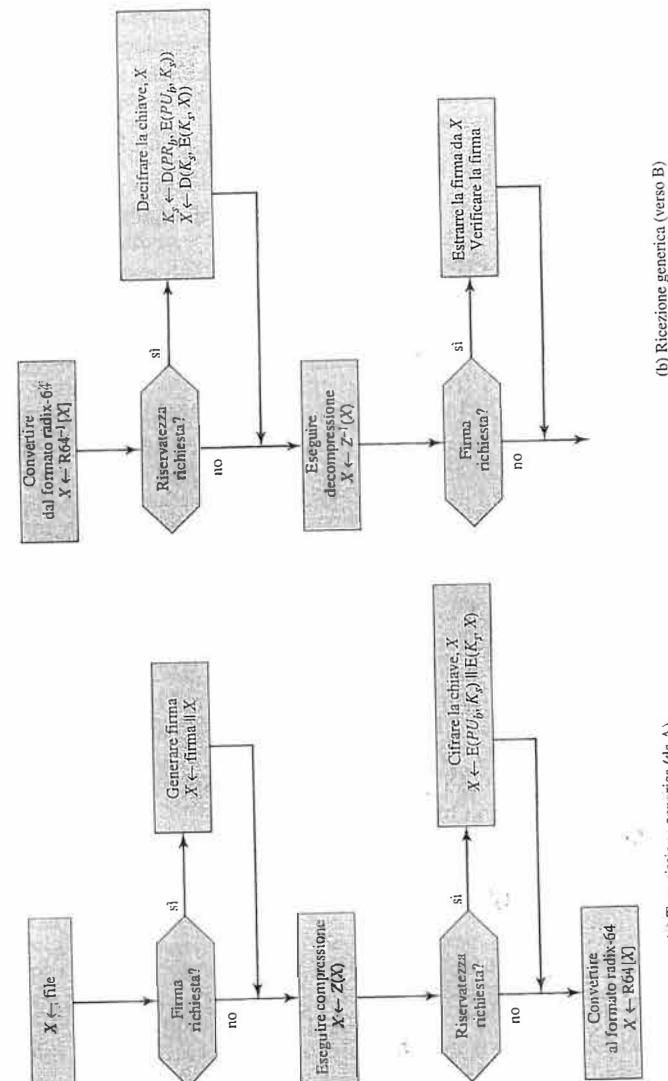


Figura 5.2 Trasmissione e ricezione di messaggi PGP.

verrà ricostruito con una chiave obsoleta. Inoltre, in questo caso, i riceventi sarebbero a conoscenza solo della vecchia chiave pubblica fino a quando non viene loro notificato l'aggiornamento. Oltre alla necessità di cambiare le chiavi, l'utente potrebbe desiderare più coppie di chiavi in un certo momento per poter interagire con gruppi diversi di corrispondenti o semplicemente per aumentare il livello di sicurezza, limitando la quantità di dati cifrata con una singola chiave. In conclusione, non esiste una corrispondenza uno-a-uno fra gli utenti e le loro chiavi pubbliche; si rende quindi necessario qualche strumento per l'identificazione di specifiche chiavi.

- Ogni entità PGP deve mantenere un file delle proprie coppie chiave pubblica/privata e un file delle chiavi pubbliche dei propri corrispondenti.

Ciascuno di questi requisiti sarà esaminato dettagliatamente di seguito.

Generazione delle chiavi di sessione

Ogni chiave di sessione è associata a un singolo messaggio ed è utilizzata solamente per la cifratura e decifratura di quel messaggio. Si rammenti che cifratura e decifratura dei messaggi sono eseguite con un algoritmo di cifratura simmetrico. CAST-128 e IDEA usano chiavi a 128 bit; 3DES usa una chiave a 168 bit.

Ipotizziamo di utilizzare CAST-128. Con questo algoritmo, vengono generati numeri casuali di 128 bit. L'ingresso al generatore di numeri casuali consiste in una chiave a 128 bit e due blocchi di 64 bit, trattati come testo in chiaro da cifrare. Utilizzando la modalità CFB (*cipher feedback*), CAST-128 produce due blocchi di testo cifrato di 64 bit, che sono concatenati per formare la chiave di sessione di 128 bit. L'algoritmo che viene usato è basato su quello specificato in ANSI X12.17.

Il testo in chiaro in ingresso al generatore di numeri casuali – che, come detto, è formato da due blocchi di 64 bit – è a sua volta derivato da un flusso di numeri casuali di 128 bit. Questi numeri sono basati su digitazioni effettuate dall'utente. Tanto il tempo di durata delle digitazioni che i valori delle digitazioni sono utilizzati per generare il flusso casuale. Quindi, se l'utente effettua digitazioni arbitrarie con il suo normale ritmo di battitura, si genera un ingresso ragionevolmente casuale. La chiave di ingresso per il generatore si ottiene combinando quest'ultimo ingresso casuale con la chiave di sessione prodotta in precedenza da CAST-128. Come risultato, data l'efficacia di CAST-128, nel rimescolamento si producono sequenze di chiavi di sessione effettivamente non prevedibili.

L'Appendice 5C descrive dettagliatamente le tecniche di generazione di numeri casuali di PGP.

Identificatori di chiave

Come precedentemente detto, un messaggio cifrato è accompagnato da una forma cifrata della chiave di sessione che era stata usata per cifrare il messaggio. La chiave di sessione stessa è cifrata mediante la chiave pubblica del ricevente. Pertanto, solo il ricevente sarà in grado di ricostruire la chiave di sessione e, quindi, il messaggio. Se ciascun utente utilizzasse una sola coppia chiave pubblica/chiave privata, il ricevente conoscerebbe automaticamente quale chiave usare per decifrare la chiave di sessione, ovvero la sua unica chiave privata. Tuttavia, è stato introdotto un requisito in base al quale ciascun utente può possedere più coppie chiave pubblica/chiave privata.

In questo caso, come può il ricevente conoscere quale delle sue chiavi pubbliche è stata utilizzata per cifrare la chiave di sessione? Una semplice soluzione potrebbe consistere nella trasmissione della chiave pubblica insieme al messaggio. Il ricevente potrebbe allora verificare che quella ricevuta è effettivamente una delle proprie chiavi pubbliche e procedere. Questo schema potrebbe funzionare, ma risulta dispendioso dal punto di vista dello spazio di memoria. Una chiave pubblica RSA può infatti avere una lunghezza di centinaia di cifre decimali. Una seconda soluzione potrebbe consistere nell'associare un identificatore – unico almeno nell'ambito delle chiavi di ciascun utente – a ciascuna chiave pubblica. In altre parole, la combinazione identificatore utente e identificatore di chiave è sufficiente a identificare univocamente una chiave. Con questa soluzione, si dovrebbe trasmettere solamente l'identificatore di chiave (molto più corto della chiave). Anche questa soluzione presenta inconvenienti riguardo alla gestione e all'occupazione dello spazio, in quanto è necessario assegnare e memorizzare gli identificatori di chiave in modo che mittente e ricevente possano mettere in corrispondenza gli identificatori con le relative chiavi. Tutto ciò sembra inutilmente oneroso.

La soluzione adottata da PGP consiste nell'assegnare a ciascuna chiave pubblica un identificatore di chiave che sia, con probabilità molto elevata, unico relativamente a un certo identificatore utente¹. L'identificatore di chiave associato a ogni chiave pubblica è formato dai 64 bit meno significativi della chiave stessa. Ovvero, l'identificatore di una chiave pubblica PU_a è $(PU_a \bmod 2^{64})$. Questa lunghezza è sufficiente a ridurre al minimo la probabilità di generare identificatori di chiave duplicati.

L'identificatore di chiave è anche necessario per la firma digitale PGP. Dato che un mittente può usare una fra le sue molteplici chiavi private per la cifratura del digest di messaggio, il ricevente deve conoscere quale chiave pubblica utilizzare. Di conseguenza, la componente firma digitale di un messaggio comprende l'identificatore di chiave di 64 bit della chiave pubblica richiesta. Alla ricezione del messaggio, il ricevente verifica dapprima che l'identificatore di chiave sia relativo a una chiave pubblica fra quelle a lui note per quel mittente e, quindi, procede alla verifica della firma.

Dopo aver introdotto il concetto di identificatore di chiave, si può procedere a un'analisi più dettagliata del formato dei messaggi trasmessi, mostrato nella Figura 5.3. Un messaggio è formato da tre componenti: la componente messaggio, la componente firma (opzionale) e la componente chiave di sessione (opzionale).

La **componente messaggio** comprende i dati da memorizzare o trasmettere, un nome di file e un valore di timestamp che specifica il tempo di creazione.

La **componente firma** comprende:

- Timestamp:** l'istante di tempo in cui è stata generata la firma.
- Digest di messaggio:** digest di 160 bit prodotto con SHA-1, cifrato con la chiave privata di firma del mittente. Il digest viene calcolato in base al valore del timestamp della firma concatenato con la porzione della componente messaggio contenente i dati. L'inclusione del valore di timestamp nel digest fornisce protezione contro attacchi di replay. L'esclusione del nome di file e del valore di timestamp della componente mes-

¹ Questa introduzione dei concetti di probabilità è stata affrontata precedentemente, nel paragrafo 8.3, per determinare se un numero è primo. Capita spesso, nella progettazione di algoritmi, che l'uso di tecniche probabilistiche porti a soluzioni che non richiedono molto tempo o che sono meno complesse, o entrambe le cose.

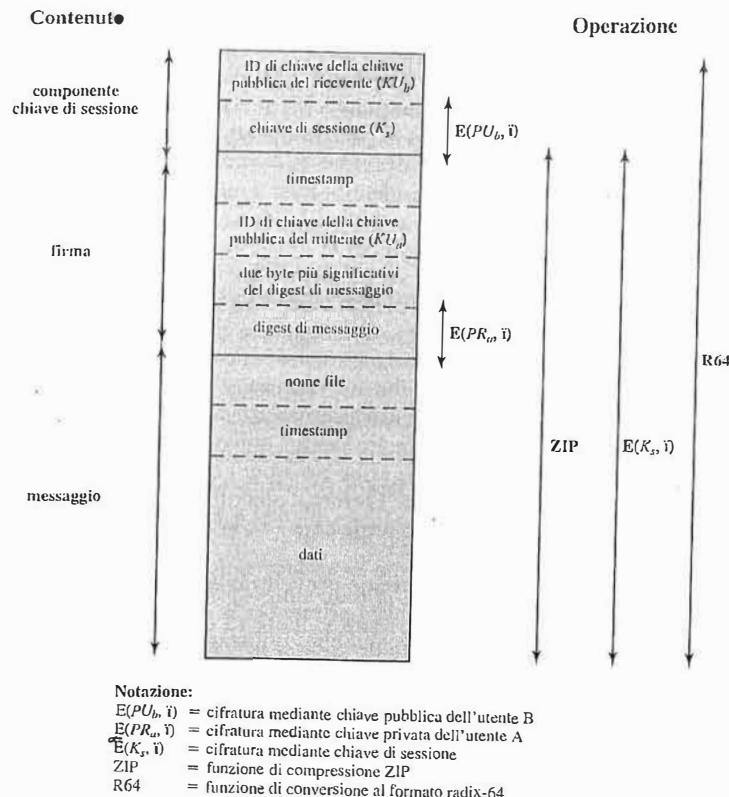


Figura 5.3 Formato generale di un messaggio PGP (da A a B).

saggio assicura che firme separate siano identiche a quelle apposte al messaggio. Le firme separate sono calcolate su un file distinto senza alcun campo di intestazione della componente messaggio.

- I due byte più significativi del digest di messaggio: consentono al ricevente di stabilire se sia stata utilizzata la chiave pubblica corretta per la decifratura del digest di messaggio per scopi di autenticazione, confrontando la copia in chiaro dei primi due byte con i primi due byte nel digest di messaggio decifrato.

Questi byte servono anche come sequenza di controllo del frame a 16 bit per il messaggio.

- Identificatore della chiave pubblica del mittente: identifica la chiave pubblica che dovrebbe essere usata per la decifratura del digest di messaggio e, quindi, identifica anche la chiave privata utilizzata per la sua cifratura.

La componente messaggio e quella opzionale di firma possono essere compresse con ZIP e possono essere cifrate con una chiave di sessione.

La componente chiave di sessione comprende la chiave di sessione e l'identificatore della chiave pubblica del ricevente, utilizzata dal mittente per cifrare la chiave di sessione.

L'intero blocco è generalmente codificato secondo il formato radix-64.

Keyring

Abbiamo appena visto come gli identificatori di chiave siano critici per il funzionamento di PGP e come un qualunque messaggio PGP comprenda due identificatori di chiave, che forniscono sia riservatezza sia autenticazione. Queste chiavi hanno necessità di essere memorizzate e organizzate in modo sistematico affinché tutte le parti coinvolte ne possano fare un uso efficace ed efficiente. Lo schema impiegato in PGP consiste nel fornire una coppia di strutture dati in ciascun nodo, una dedicata alla memorizzazione delle coppie chiave pubblica/chiave privata possedute dal nodo e l'altra dedicata alla memorizzazione delle chiavi pubbliche degli altri utenti noti al nodo. Queste strutture dati sono chiamate rispettivamente keyring (portachiavi) privato e keyring pubblico.

La struttura generale del **keyring privato** è mostrata nella Figura 5.4. Si può considerare il keyring come una tabella, in cui ogni riga rappresenta una coppia chiave pubblica/chiave privata posseduta dall'utente in questione. Ogni riga ha le seguenti voci:

- timestamp:** la data/istante di tempo in cui la coppia di chiavi in questione è stata generata;

Keyring privato

Timestamp	ID* di chiave	Chiave pubblica	Chiave privata cifrata	ID* utente
T	T	T	T	T
T	T	T	T	T
T	T	T	T	T
T _i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	Utente i
T	T	T	T	T
T	T	T	T	T

Keyring pubblico

Timestamp	ID* di chiave	Chiave pubblica	Livello di fiducia del proprietario	ID* utente	Legittimità della chiave	Firma/e	Livello/i di fiducia delle firme
T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T
T _i	$PU_i \bmod 2^{64}$	PU_i	$trust_flag_i$	Utente i	$trust_flag_i$		
T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T

* = Campo utilizzato per l'indicizzazione della tabella

Figura 5.4 Struttura generale del keyring privato e del keyring pubblico.

- **identificatore di chiave:** i 64 bit meno significativi della chiave pubblica per la voce considerata;
- **chiave pubblica:** la chiave pubblica della coppia;
- **chiave privata:** la chiave privata della coppia (questo campo è cifrato);
- **identificatore utente:** tipicamente è l'indirizzo di posta elettronica dell'utente (ad esempio, stallings@acm.org). Tuttavia, l'utente può scegliere di associare un nome diverso con ciascuna coppia (come: Stallings, WStallings, WilliamStallings) oppure riutilizzare lo stesso identificatore utente più di una volta.

Anche se il keyring privato è destinato ad essere memorizzato solo sulla macchina dell'utente che ha creato e possiede le copie di chiavi, ed è accessibile solo a quell'utente, è ragionevole fare in modo che il valore della chiave privata sia il più possibile sicuro. Di conseguenza, la chiave privata vera e propria non è memorizzata nel corrispondente keyring. Al contrario, viene cifrata con CAST-128 (o con IDEA oppure 3DES). La procedura opera come segue.

1. L'utente seleziona una passphrase da usare per la cifratura delle chiavi private.
2. Quando il sistema genera una nuova coppia chiave pubblica/chiave privata con RSA, chiede all'utente la passphrase. Viene quindi generato un codice hash di 160 bit con SHA-1, sulla base della passphrase fornita dall'utente, e la passphrase viene eliminata.
3. Il sistema effettua la cifratura della chiave privata con l'algoritmo CAST-128, utilizzando come chiave i 128 bit della codifica hash. La chiave privata cifrata viene memorizzata nel keyring privato.

Successivamente, un utente deve fornire la passphrase quando accede al keyring privato per reperire una chiave privata. PGP reperirà la chiave privata cifrata, genererà il codice hash con la passphrase e decifrerà la chiave cifrata con l'algoritmo CAST-128 e il codice hash generato.

Questo schema risulta molto compatto ed efficace. Come in tutti i sistemi basati su password, la sicurezza del sistema dipende dalla sicurezza delle password. Per evitare la tentazione di trascriverla, l'utente dovrebbe scegliere una passphrase facile da ricordare ma non da indovinare.

La struttura generale del **keyring pubblico** è anch'essa mostrata nella Figura 5.4. Questa struttura dati viene utilizzata per memorizzare le chiavi pubbliche degli altri utenti conosciuti dall'utente in questione. Tralasciamo per il momento alcuni campi della tabella nella Figura 5.4, limitandoci alla descrizione dei seguenti:

- **timestamp:** data/istante temporale di generazione della voce considerata;
- **identificatore di chiave:** i 64 bit meno significativi della chiave pubblica per la voce considerata;
- **chiave pubblica:** per la voce considerata;
- **identificatore utente:** identifica il possessore della chiave in questione. Si possono associare molteplici identificatori utente a una singola chiave pubblica.

Sia il keyring pubblico sia quello privato possono essere indicizzati, in alternativa, o per identificatore utente o per identificatore di chiave; successivamente si esamineranno le motivazioni per entrambe le modalità di indicizzazione.

Siamo ora in grado di mostrare come vengono utilizzati i keyring in fase di trasmissione e ricezione dei messaggi. Per semplicità, ignoreremo le operazioni di compressione e di conversione nel formato radix-64. Dapprima si considera la trasmissione dei messaggi (Figura 5.5), ipotizzando che questi debbano essere sia firmati sia cifrati. Il mittente PGP esegue i seguenti passi.

- 1. Firma del messaggio.**
 - a. PGP reperisce la chiave privata del mittente dal keyring privato utilizzando il valore di identificatore `your_userid` come indice. Se non viene fornito alcun valore `your_userid` nel comando, viene reperita la prima chiave privata fra quelle nel keyring.
 - b. PGP chiede all'utente la passphrase per ricostruire la chiave privata in chiaro.
 - c. Si costruisce la componente firma del messaggio.
- 2. Cifratura del messaggio.**
 - a. PGP genera una chiave di sessione ed effettua la cifratura del messaggio.
 - b. PGP reperisce la chiave pubblica del ricevente dal keyring pubblico utilizzando il valore `her_userid` come indice.
 - c. Si costruisce la componente chiave di sessione del messaggio.

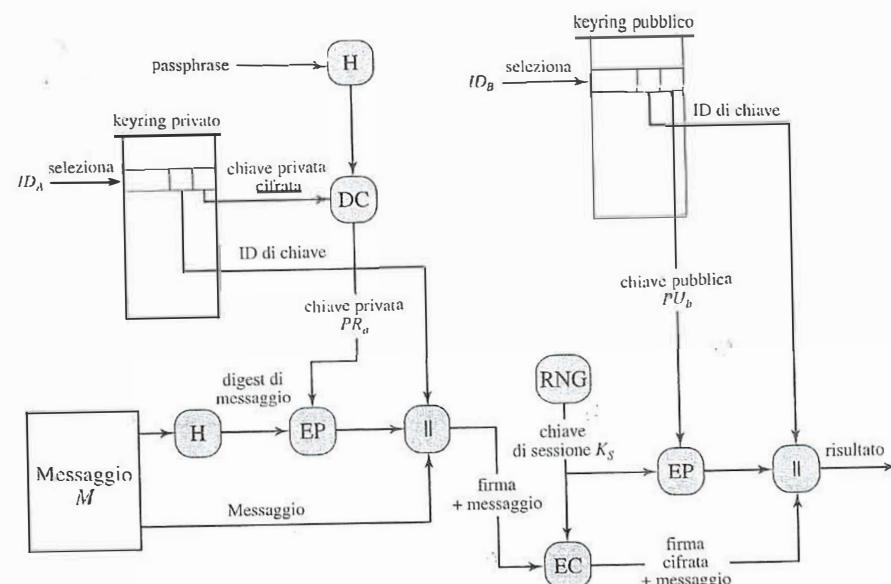


Figura 5.5 Generazione di un messaggio PGP (da un utente A a un utente B, senza compressione o conversione in radix-64).

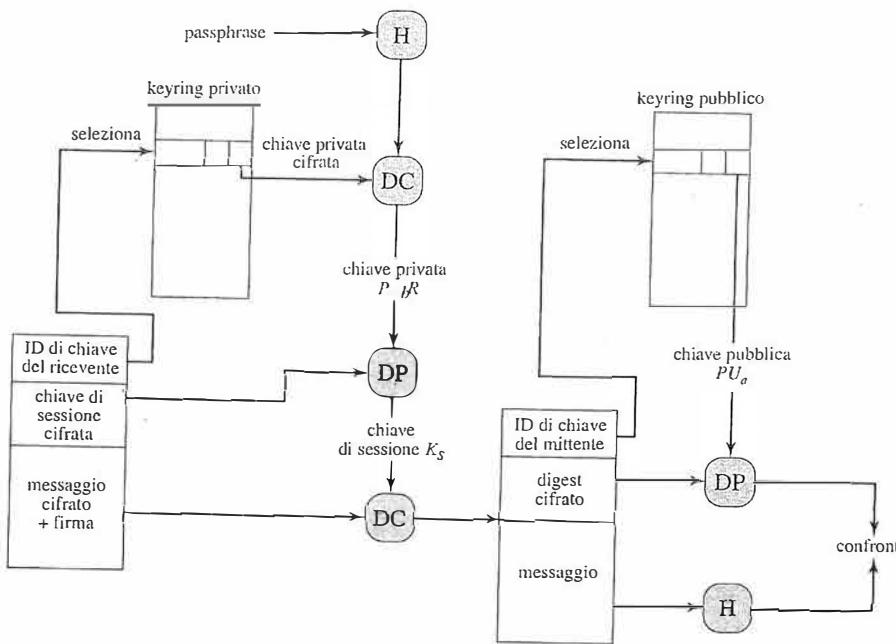


Figura 5.6 Ricezione di un messaggio PGP (da un utente A ad un utente B, senza compressione o conversione in radix-64).

Il ricevente PGP esegue i seguenti passi (Figura 5.6).

1. Decifratura del messaggio.

- PGP reperisce la chiave privata del ricevente dal keyring privato, utilizzando come indice il campo identificatore di chiave della componente chiave di sessione contenuta nel messaggio.
- PGP chiede all'utente la passphrase per ricostruire la chiave privata in chiaro.
- PGP ricostruisce a questo punto la chiave di sessione ed effettua la decifratura del messaggio.

2. Autenticazione del messaggio.

- PGP reperisce la chiave pubblica del mittente dal keyring pubblico, usando come indice il campo identificatore di chiave nella componente firma del messaggio.
- PGP ricostruisce il digest di messaggio trasmesso.
- PGP calcola il digest di messaggio per il messaggio ricevuto e lo confronta con quel b trasmesso, per l'autenticazione.

Gestione delle chiavi pubbliche

Come emerso dalla precedente descrizione, PGP contiene un ingegnoso ed efficiente insieme di funzioni e formati indipendenti, per fornire un efficace servizio di riservatezza e autenticazione. Per completare il sistema, occorre esaminare ancora la gestione delle chiavi pubbliche. Anche la documentazione PGP riconosce l'importanza di questo argomento.

PGP fornisce una struttura per risolvere questo problema, suggerendo alcune possibili opzioni. Dato che PGP è destinato ad essere utilizzato in molteplici ambienti, con requisiti di sicurezza diversificati, non prevede uno schema rigido di gestione delle chiavi, come nel caso di S/MIME, descritto in seguito.

Approcci alla gestione delle chiavi pubbliche

Il cuore del problema è questo: un utente A deve costruire un keyring pubblico contenente le chiavi pubbliche degli altri utenti al fine di poter scambiare messaggi mediante PGP. Supponiamo che il keyring di A contenga una chiave pubblica attribuita a B ma che in realtà è una chiave di C. Questo potrebbe accadere, ad esempio, se A scaricasse la chiave da un *bulletin board system* (BBS), usato da B per collocare la propria chiave pubblica ma compromessa da C. Come risultato, si prospettano due possibili rischi. Innanzitutto, C può spedire messaggi ad A falsificando la firma di B, e quindi A acetterà il messaggio come se provenisse da B. Secondariamente, un qualunque messaggio cifrato inviato da A a B può essere letto da C.

Sono possibili diversi approcci per minimizzare il rischio che i keyring pubblici contengano chiavi false. Si supponga che A voglia ottenere una chiave pubblica affidabile per B. Ecco alcuni tra gli approcci utilizzabili.

- A riceve fisicamente la chiave da B. B potrebbe memorizzare la propria chiave pubblica (PU_b) su un dischetto e consegnarlo a mano ad A che potrebbe usarlo per caricare la chiave nel proprio sistema. Questo è un metodo molto sicuro ma ha ovvie limitazioni di ordine pratico.
- A verifica la chiave telefonicamente. Se è in grado di riconoscere B al telefono, A potrebbe chiamarlo per farsi dettare la chiave in formato radix-64 al telefono. Come alternativa più pratica, B potrebbe trasmettere la propria chiave ad A in un messaggio di posta elettronica. Dal canto suo, A potrebbe generare con PGP un digest di 160 bit della chiave tramite SHA-1 e visualizzarlo in formato esadecimale: questo metodo è conosciuto come "impronta digitale" della chiave. A potrebbe poi telefonare a B e chiedere la dettatura dell'impronta digitale al telefono. Se le due impronte combaciano, la chiave è verificata.
- A ottiene la chiave di B da un'altra persona, D, di cui entrambe le parti di fidano. A tale scopo, D agisce da *introducer* (certificatore che firma la chiave di B) creando un certificato firmato. Il certificato comprende la chiave pubblica di B, il tempo di creazione della chiave e il periodo di validità della chiave stessa. D genera anche un digest SHA-1 di questo certificato, ne esegue la cifratura con la propria chiave privata e lo appone come firma al certificato. Poiché solo D può aver creato la firma, nessun altro può generare una chiave pubblica falsa fingendo che sia firmata da D. Il certificato firmato può essere spedito direttamente ad A da B o da D, oppure può essere pubblicato su un BBS.

4. A ottiene la chiave pubblica di B da un'autorità certificativa fidata. Anche in questo caso viene creato un certificato firmato dall'autorità. A potrebbe quindi fornire il nominativo di un utente all'autorità e ricevere il corrispondente certificato firmato.

Nei casi 3 e 4, A dovrebbe essere già in possesso di una copia della chiave pubblica della terza parte che agisce da introducer e fidarsi del fatto che sia valida. In definitiva, è facoltà di A attribuire un livello di fiducia all'entità che agisce da introducer.

Utilizzo dei livelli di fiducia

Anche se PGP non comprende specifiche per stabilire autorità certificativa o per stabilire il livello di fiducia, fornisce comunque un valido strumento per utilizzare livelli di fiducia, associare livelli di fiducia alle chiavi pubbliche e sfruttare informazioni sui livelli di fiducia.

La struttura di base è la seguente. Come precedentemente detto, ciascuna voce del keyring pubblico è un certificato a chiave pubblica. A ciascuna voce è associato un **campo di legittimità della chiave**, che specifica il grado di fiducia di PGP nel fatto che la chiave in questione sia una chiave pubblica valida per quell'utente; tanto più elevato è il livello di fiducia, tanto più forte è il legame fra l'identificatore utente e la chiave in questione. Il campo di legittimità è calcolato da PGP. Inoltre, a ciascuna voce sono anche associate zero o più firme, raccolte dal proprietario del keyring, per la firma del certificato in questione. Ciascuna firma, a sua volta, ha associato un **campo di fiducia della firma**, che specifica il grado di fiducia, che l'utente PGP in questione ripone nell'entità firmataria per la certificazione delle chiavi pubbliche. Il campo di legittimità della chiave di una certa voce è derivato dall'insieme dei campi di fiducia della firma associati alla voce stessa. Infine, ciascuna voce definisce una chiave pubblica associata a un particolare proprietario, e comprende un **campo di fiducia del proprietario**, che specifica quanto fidata è da considerarsi la chiave in questione per la firma di altri certificati; questo livello di fiducia è assegnato dall'utente. Si può pensare ai campi di fiducia della firma come a copie, memorizzate in cache, del campo di fiducia del proprietario da un'altra voce.

Questi tre campi sono contenuti in una struttura denominata *trust flag byte*. La Tabella 5.2 mostra il contenuto del trust flag byte per ciascuno dei tre campi. Si supponga di considerare il keyring pubblico dell'utente A. Il processo di attribuzione dei livelli di fiducia opera come segue.

- Quando A inserisce una nuova chiave pubblica nel keyring pubblico, PGP deve attribuire un valore al campo di fiducia associato al proprietario di questa chiave. Se il proprietario è A, e pertanto la chiave in questione compare anche nel keyring privato, al campo di fiducia viene attribuito automaticamente il valore di **massima fiducia (ultimate trust)**. Altrimenti, PGP chiede ad A di stabilire il livello di fiducia da attribuire al proprietario della chiave in questione, e A inserisce il livello desiderato. L'utente può specificare un valore tra: sconosciuto, non fidato, marginalmente fidato o completamente fidato.
- Quando la nuova chiave pubblica viene inserita, questa può avere allegate una o più firme. Ulteriori firme possono essere aggiunte successivamente. Quando una firma viene aggiunta alla voce in questione, PGP effettua una ricerca nel keyring pubblico per verificare se l'autore della firma è presente fra i proprietari di chiavi pubbliche co-

(a) Livello di fiducia attribuito al proprietario della chiave pubblica (compare dopo il pacchetto contenente la chiave definito dall'utente)	(b) Livello di fiducia attribuito alla coppia chiave pubblica/identificatore utente (compare dopo il pacchetto contenente l'identificatore utente; calcolato da PGP)	(c) Livello di fiducia attribuito alla firma (compare dopo il pacchetto contenente la firma; copia di OWNERTRUST per il firmatario corrente)
Campo OWNERTRUST	Campo KEYID [†]	Campo SIGTRUST
<ul style="list-style-type: none"> — fiducia non nota o non definita — proprietario della chiave: non fidato — utente sconosciuto — normalmente non fidato per la firma di altre chiavi — normalmente fidato per la firma di altre chiavi — sempre fidato per la firma di altre chiavi — questa chiave è presente nel keyring segreto (massima fiducia) 	<ul style="list-style-type: none"> — fiducia non definita — proprietario della chiave: non fidato — fiducia parziale nel proprietario della chiave — fiducia completa nel proprietario della chiave 	<ul style="list-style-type: none"> — utente sconosciuto — normalmente non fidato per la firma di altre chiavi — normalmente fidato per la firma di altre chiavi — sempre fidato per la firma di altre chiavi — questa chiave è presente nel keyring segreto (massima fiducia)
Bit WARNONLY		Bit CONTIG
	<ul style="list-style-type: none"> — posto a 1 se l'utente desidera solo essere avvisato quando una chiave non completamente fidata viene usata per la cifratura 	<ul style="list-style-type: none"> — posto a 1 se la firma consente un cammino di certificazione fidato contiguo che torna fino al proprietario del suddetto keyring di chiavi
Bit BUCKSTOP		
	<ul style="list-style-type: none"> — posto a 1 se la chiave corrispondente compare nel keyring segreto 	

Tabella 5.2 Contenuti del trust flag byte.

nosciuti. In caso affermativo, si assegna il valore OWNERTRUST di questo proprietario al campo SIGTRUST per la firma in questione. In caso negativo, si assegna il valore **utente sconosciuto (unknown user)**.

3. Il valore del campo di legittimità della chiave è calcolato sulla base dei campi di fiducia della firma presenti nella voce corrente. Se almeno una firma possiede un valore di fiducia pari a **massima fiducia (ultimate trust)**, allora si pone a **completo (complete trust)** il valore di legittimità della chiave. In caso contrario, PGP calcola una somma pesata dei valori di fiducia. Si assegna peso pari a $1/X$ alle firme che hanno valore **sempre fidato (always trusted)** e un peso pari a $1/Y$ alle firme che hanno valore **generalmente fidato (usually trusted)**, dove X e Y sono parametri configurabili dall'utente. Se la somma dei pesi raggiunge 1, il valore del campo di legittimità della chiave è posto a **completo (complete trust)**. Quindi, se nessuna firma possiede un valore di fiducia pari a massima fiducia (*ultimate trust*), sono necessarie almeno X firme con valore sempre fidato (*always trusted*) o Y firme con valore generalmente fidato (*usually trusted*), o una loro combinazione.

Periodicamente, PGP elabora il keyring pubblico per garantirne la consistenza. Si tratta in sostanza di un processo top-down. Per ciascun campo OWNERTRUST, PGP scandisce il keyring verificando tutte le firme certificate da quel proprietario e aggiorna il campo SIGTRUST allo stesso valore di OWNERTRUST. Questo processo inizia con le chiavi aventi valore massima fiducia (*ultimate trust*). Quindi si calcolano tutti i campi KEYLEGIT sulla base delle firme associate.

La Figura 5.7 mostra un esempio della correlazione tra i parametri di fiducia delle firme e di legittimità delle chiavi², esemplificando la struttura di un keyring pubblico. L'utente ha acquisito un certo numero di chiavi pubbliche, alcune direttamente dai loro proprietari e altre da una terza parte come, ad esempio, un server di chiavi, ovvero server predisposti alla pubblicazione di chiavi pubbliche certificate.

Il nodo etichettato "Utente U" fa riferimento alla voce del keyring pubblico corrispondente all'utente U. Questa chiave è legittima e il valore di OWNERTRUST è pari a massima fiducia (*ultimate trust*). Ogni altro nodo nel keyring ha un valore di OWNERTRUST pari a **non definito (undefined trust)**, a meno che l'utente non abbia specificato un valore diverso. Nell'esempio, l'utente in questione ha specificato che si fiderà sempre degli utenti D, E, F e L per la firma di altre chiavi (valore *always trusted*). Sempre per la firma di altre chiavi, l'utente si fida solo parzialmente degli utenti A e B (valore *usually trusted*).

Le diverse ombreggiature dei nodi nella Figura 5.7 indicano il livello di fiducia attribuito dall'utente U. La struttura ad albero indica quali chiavi sono state firmate e da quali altri utenti. Se una chiave è stata firmata da un utente la cui chiave è anch'essa presente nel keyring in questione, la freccia connette la chiave firmata al firmatario. Se una chiave è stata firmata da un utente la cui chiave non è presente nel keyring in questione, la freccia connette la chiave firmata a un punto interrogativo che denota il fatto che il firmatario della chiave è sconosciuto all'utente considerato.

La Figura 5.7 consente alcune considerazioni.

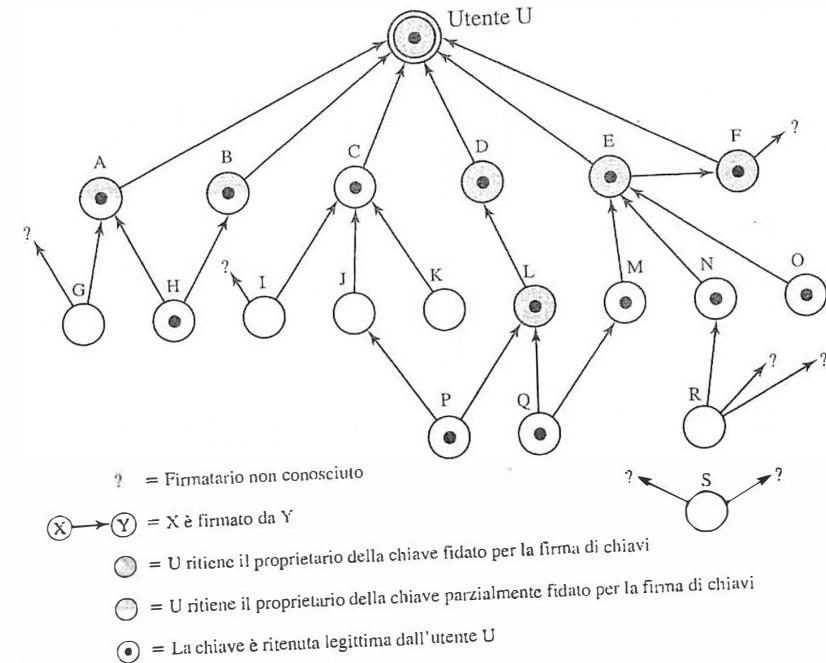


Figura 5.7 Esempio di livelli di fiducia di PGP.

1. Si può notare che l'utente in questione nell'esempio ha firmato tutte le chiavi i cui proprietari godono completamente o parzialmente della sua fiducia, ad eccezione del nodo L. Questa firma non è sempre necessaria (come dimostrato dal nodo L), anche se la maggior parte degli utenti generalmente firmerà le chiavi della maggior parte degli utenti di cui ha fiducia. Ad esempio, anche se la chiave di E è già firmata dall'introduttore fidato F, l'utente in questione sceglie di firmare comunque la chiave di E.
2. Si ipotizza che due firme parzialmente fidate siano sufficienti a certificare una chiave. Quindi, PGP considera legittima la chiave dell'utente H in quanto firmata da A e B, ciascuno dei quali è parzialmente fidato.
3. Si può ritenere una chiave legittima perché è firmata da un utente completamente fidato o da due firmatari parzialmente fidati, ma l'utente associato a quella chiave potrebbe non essere considerato fidato per la firma di altre chiavi. Ad esempio, la chiave di N è legittima perché firmata da E, di cui l'utente U si fida, ma N non è fidato per la firma di altre chiavi, in quanto l'utente U non ha attribuito a N tale livello di fiducia. Pertanto, anche se la chiave di R è firmata da N, PGP non considera legittima la chiave di R. Questa situazione trova una sua spiegazione. Se si desidera inviare un messaggio privato a una persona, non è necessario fidarsi completamente di quella persona ma è sufficiente essere sicuri di possedere la chiave pubblica corretta.

² La figura è stata fornita all'autore da Phil Zimmermann.

4. La Figura 5.7 mostra anche un esempio di nodo “orfano” staccato (nodo S), con due firme di utenti sconosciuti. Tale chiave potrebbe essere stata acquisita da un server di chiavi. PGP non può ipotizzare che tale chiave sia legittima semplicemente perché proveniente da un server rispettabile. L’utente deve dichiarare la legittimità della chiave firmandola o segnalando a PGP di essere intenzionato a fidarsi completamente di uno dei firmatari della chiave.

Un’ultima considerazione: prima è stato detto che una singola chiave pubblica sul keyring pubblico può essere associata a più identificativi utente. Ciò può capitare perché una persona ha cambiato nome o è stata inserita tramite la firma sotto più nomi indicando, ad esempio, diversi indirizzi di posta elettronica per la stessa persona. Quindi una chiave pubblica può essere vista come la radice di un albero. Questa ha un certo numero di identificatori utente associati, con un certo numero di firme per ciascun identificatore utente. La corrispondenza fra un particolare identificatore utente e una chiave dipende dalle firme associate a quell’identificatore utente e a quella chiave, mentre il livello di fiducia nella chiave (per la firma di altre chiavi) è funzione di tutte le firme associate.

Revoca delle chiavi pubbliche

Un utente può voler revocare la propria chiave pubblica correntemente in uso in quanto sospetta una compromissione o semplicemente per evitare di utilizzarla troppo a lungo. Si noti che la compromissione richiederebbe che un avversario abbia ottenuto in qualche modo una copia in chiaro della chiave privata oppure abbia ottenuto sia la chiave privata dal keyring privato sia la passphrase.

La convenzione per la revoca di una chiave pubblica consiste nell’emissione, da parte del proprietario della chiave, di un certificato di revoca da lui firmato. Questo certificato ha lo stesso formato di un usuale certificato di firma, ma include anche un indicatore che specifica che il certificato ha lo scopo di revocare una chiave pubblica. Si noti che per firmare un certificato di revoca di chiave pubblica occorre usare la corrispondente chiave privata. Il proprietario dovrebbe quindi tentare di diffondere il certificato di revoca quanto più ampiamente e rapidamente possibile al fine di consentire ai suoi potenziali corrispondenti di aggiornare i loro keyring pubblici.

In effetti, anche un avversario che abbia compromesso la chiave privata di un proprietario può emettere tale certificato. Tuttavia, questo impedirebbe l’uso della chiave pubblica all’avversario sia al proprietario legittimo; vi è quindi una minor probabilità che questa situazione si verifichi rispetto all’uso doloso che un avversario potrebbe fare di una chiave privata rubata.

5.2 S/MIME

S/MIME (*secure/multipurpose Internet mail extensions*) costituisce un miglioramento alla sicurezza del formato standard di posta elettronica Internet MIME, basato su tecnologia di RSA Data Security. Anche se PGP e S/MIME sono standard IETF, sembra probabile che S/MIME emergerà come standard industriale per uso commerciale e aziendale, mentre PGP si configurerà come strumento per la sicurezza della posta elettronica personale. S/MIME è definito in parecchi documenti, le RFC più importanti sono 3369, 3370, 3850 e 3851.

Per poter comprendere S/MIME, è necessario prima di tutto capire il sottostante formato di posta elettronica utilizzato, ovvero MIME, risalendo fino al tradizionale formato di posta elettronica standard, RFC 822, ancora comunemente utilizzato. Il paragrafo, quindi, introduce in primo luogo questi due standard per poi affrontare la trattazione specifica di S/MIME.

RFC 822

RFC 822 definisce un formato per i messaggi di testo inviati mediante posta elettronica. È stato lo standard per i messaggi di testo di posta su Internet ed è tuttora di comune utilizzo. In ambito RFC 822, i messaggi sono considerati come se fossero costituiti da un involucro (busta) e un contenuto; il primo comprende tutte le informazioni necessarie per effettuare la trasmissione e la consegna; il secondo rappresenta l’oggetto da recapitare al destinatario. Lo standard RFC 822 si applica solo al contenuto. Lo standard del contenuto, inoltre, include un insieme di campi intestazione che possono essere utilizzati dal sistema di posta per creare l’involucro; questo standard ha lo scopo di facilitare l’acquisizione di queste informazioni da parte dei programmi.

La struttura complessiva di un messaggio conforme allo standard RFC 822 è molto semplice. Un messaggio consiste in un certo numero di righe di intestazione (*intestazione*) seguite da un testo senza limitazioni (*corpo*). L’intestazione è separata dal corpo da una riga vuota. In altri termini, un messaggio è un testo ASCII, e tutte le righe fino alla prima riga vuota costituiscono l’intestazione e sono utilizzate dalla parte agente dell’utente del sistema di posta.

Una linea di intestazione generalmente contiene una parola chiave, seguita da due punti che introducono gli argomenti della parola chiave; il formato consente di suddividere una riga di lunghezza elevata in un certo numero di righe più brevi. Le parole chiave più frequentemente usate sono *From*, *To*, *Subject*, e *Date*. Ecco un esempio di messaggio:

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com
```

Ciao. Questa sezione
dà inizio al corpo del messaggio
ed è separato dall’intestazione da una riga vuota.

Un altro campo che si trova comunemente nelle intestazioni RFC 822 è *Message-ID*, che contiene un identificatore unico associato al messaggio.

MIME

MIME è un'estensione di RFC 822 che si prefigge di superare alcune delle problematiche e limitazioni nell'uso di SMTP (*simple mail transfer protocol*) o di alcuni altri protocolli di trasferimento della posta e di RFC 822 per la posta elettronica. [RODR02] individua le seguenti limitazioni dello schema SMTP/822.

1. SMTP non può trasmettere file eseguibili o altri oggetti binari. Esistono numerosi schemi per la conversione di file binari in formato testo che possono essere utilizzati dai sistemi di posta SMTP, fra cui il popolare schema UNIX UUencode/UUdecode. Nessuno di questi tuttavia è uno standard o uno standard *de facto*.
2. SMTP non può trasmettere dati testuali contenenti caratteri peculiari di un particolare alfabeto nazionale in quanto rappresentati da codifiche a 8 bit con valori pari al valore 128 decimale o superiori, mentre SMTP si limita a codifiche ASCII a 7 bit.
3. I server SMTP possono rifiutare messaggi di posta che superino una certa dimensione.
4. I gateway SMTP che effettuano la conversione fra ASCII e codifica di carattere EBCDIC hanno alcuni problemi di conversione, in quanto non usano un insieme consistente di corrispondenze.
5. I gateway SMTP verso reti X.400 di posta elettronica non possono gestire dati non testuali contenuti nei messaggi X.400.
6. Alcune implementazioni SMTP non sono completamente conformi agli standard SMTP definiti nella RFC 821. Fra i problemi più comuni si hanno:
 - cancellare, aggiungere, o risistemare i caratteri di invio e di riempimento riga (*linefeed*);
 - troncare le righe superiori ai 76 caratteri;
 - rimuovere spaziature (tabulazioni e caratteri di spazio);
 - completare le righe in un messaggio per riportarle tutte a una stessa lunghezza;
 - convertire le tabulazioni in molteplici caratteri di spazio.

MIME si prefigge di risolvere questi problemi in maniera compatibile con le implementazioni esistenti di RFC 822. La specifica è fornita nelle RFC dalla 2045 alla 2049.

Descrizione generale

La specifica MIME comprende i seguenti elementi.

1. Sono definiti cinque nuovi campi di intestazione dei messaggi che possono comparire all'interno di un'intestazione RFC 822. Questi campi forniscono informazioni relative al corpo del messaggio.
2. Sono definiti un certo numero di formati per il contenuto, standardizzando quindi le rappresentazioni che supportano posta elettronica multimediale.

3. Sono definite codifiche di trasferimento che rendono possibile la conversione di un qualunque formato del contenuto in un formato protetto da modifiche da parte del sistema di posta.

Di seguito vengono introdotti prima i cinque campi di intestazione e quindi i formati del contenuto e le codifiche di trasferimento.

I cinque campi di intestazione definiti in MIME sono i seguenti.

- **MIME-Version:** deve avere il valore 1.0. Questo campo indica che il messaggio è conforme alle RFC 2045 e 2046.
- **Content-Type:** descrive i dati contenuti nel corpo del messaggio con un dettaglio sufficiente a consentire all'agente utente ricevente di selezionare un agente o un meccanismo appropriato per la rappresentazione dei dati all'utente o, comunque, di trattare i dati in maniera appropriata.
- **Content-Transfer-Encoding:** indica il tipo di trasformazione utilizzata per rappresentare il corpo del messaggio in maniera adatta al trasporto sotto forma di posta.
- **Content-ID:** utilizzato per identificare univocamente entità MIME in molteplici contesti.
- **Content-Description:** descrizione testuale dell'oggetto associato al corpo; utile quando l'oggetto non è leggibile (per esempio, dati audio).

Uno, alcuni o tutti questi campi possono apparire in una comune intestazione RFC 822. Un'implementazione compatibile deve supportare i campi MIME-Version, Content-Type e Content-Transfer-Encoding; i campi Content-ID e Content-Description sono opzionali e possono essere ignorati dall'implementazione del ricevente.

Tipi di contenuto di MIME

La parte più consistente della specifica MIME riguarda la definizione di una varietà di possibili tipi di contenuto. Ciò riflette la necessità di fornire modalità standardizzate per il trattamento di una vasta gamma di rappresentazioni dell'informazione in ambiente multimediale.

La Tabella 5.3 fornisce un elenco dei tipi di contenuto specificati nella RFC 2046. Esistono sette diversi tipi principali di contenuto e un totale di 15 sottotipi. In generale, un tipo di contenuto specifica il tipo generale di dati, mentre il sottotipo specifica un particolare formato per quel tipo di dati.

Il tipo **text** del corpo non richiede software speciale per l'acquisizione del significato del testo, a parte il supporto per l'insieme di caratteri specificato. Il sottotipo principale è *plain text*, che consiste semplicemente in una stringa di caratteri ASCII o ISO 8859. Il sottotipo *enriched* consente una maggiore flessibilità di formattazione.

Il tipo **multipart** indica che il corpo contiene più parti indipendenti. Il campo di intestazione Content-Type comprende un parametro, denominato *boundary* (confine), che definisce la delimitazione tra le parti che costituiscono il corpo. Il boundary non deve comparire in alcuna parte del messaggio. Ciascun boundary inizia su una nuova riga ed è costituito da due trattini seguiti dal valore di boundary. L'ultimo boundary, che indica la fine dell'ultima parte, ha un ulteriore suffisso di due trattini. All'interno di ciascuna parte, ci può essere una normale intestazione MIME (opzionale). *

Tipo	Sottotipo	Descrizione
Text	Plain	Testo non formattato; può essere ASCII o ISO 8859.
	Enriched	Fornisce maggiore flessibilità di formato.
Multipart	Mixed	Le diverse parti sono indipendenti ma devono essere trasmesse insieme. Dovrebbero essere presentate al ricevente nell'ordine con cui appaiono nel messaggio di posta.
	Parallel	Differisce da Mixed solo per il fatto che non è definito un ordine con cui recapitare le parti al ricevente.
	Alternative	Le diverse parti sono versioni alternative della stessa informazione. Sono disposte in ordine crescente di somiglianza rispetto all'originale e il sistema di posta del ricevente dovrebbe mostrare all'utente la versione "migliore".
	Digest	Simile a Mixed, ma il tipo/sottotipo di default di ogni parte è message/rfc822.
Message	rfc822	Il corpo è anch'esso un messaggio encapsulato conforme a RFC 822.
	Partial	Usato per consentire la scomposizione di messaggi di posta di dimensioni elevate, in maniera trasparente al ricevente.
	External-body	Contiene un puntatore a un oggetto che esiste altrove.
Image	jpeg	L'immagine è in formato JPEG, con codifica JIFIE.
	gif	L'immagine è in formato GIF.
Video	mpeg	Formato MPEG.
Audio	Basic	Codifica mono-canalé ISDN, quantizzazione 8 bit mu-law, frequenza di campionamento 8 kHz.
Application	PostScript	Postscript Adobe.
	octet-stream	Dati binari di tipo generico consistenti di byte.

Tabella 5.3 Tipi di contenuto MIME.

Ecco un esempio di messaggio multipart, contenente due parti, a loro volta costituite da testo semplice (tratto dalla RFC 2046):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"
Questo e' il preambolo. Va ignorato, tuttavia e uno spazio utile
in cui inserire note esplicative per lettori che non utilizzano
MIME.
--simple boundary
Questo e' puro testo ASCII non esplicitamente digitato. NON termina
con un'interruzione di riga.
--simple boundary
Content-Type: text/plain; charset=us-ascii
Questo e' puro testo ASCII digitato esplicitamente. DEVE terminare
con un'interruzione di riga.
--simple boundary--
Questo e' l'epilogo. Anch'esso va ignorato.
```

Tutti i quattro sottotipi del tipo multipart hanno la stessa sintassi complessiva. Il **sottotipo multipart/mixed** viene usato in presenza di molte parti indipendenti del corpo che devono essere collegate secondo un ordine particolare. Nel **sottotipo multipart/parallel**, l'ordine delle parti non è significativo. Se il sistema del ricevente è adatto, le diverse parti possono essere presentate in parallelo. Ad esempio, un'immagine o un testo possono essere accompagnati da un commento vocale, che viene fatto ascoltare durante la visualizzazione dell'immagine o del testo.

Nel **sottotipo multipart/alternative**, le parti sono una diversa rappresentazione della stessa informazione. Eccone un esempio.

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Posta testuale formattata
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary="boundary42"
--boundary42
Content-Type: text/plain; charset=us-ascii
...la versione ASCII del messaggio si trova qui...
--boundary42
Content-Type: text/enriched
...la versione RFC 1896 text/enriched dello stesso messaggio si
trova qui...
--boundary42--
```

In questo sottotipo, le parti del corpo sono ordinate in termini di preferenza crescente. Nell'esempio precedente, se il sistema del ricevente è in grado di mostrare il messaggio in formato text/enriched, ciò viene effettuato; altrimenti, si usa il formato ASCII.

Il **sottotipo multipart/digest** viene usato quando ciascuna delle parti del corpo è interpretata come un messaggio RFC 822 con intestazioni. Questo sottotipo consente la co-

struzione di un messaggio le cui parti sono singoli messaggi. Ad esempio, il moderatore di un gruppo potrebbe raccogliere messaggi di posta elettronica dei partecipanti al gruppo, metterli insieme e inviarli all'interno di un messaggio MIME che li incapsula.

Il tipo **message** fornisce un certo numero di funzionalità importanti di MIME. Il sottotipo **message/rfc822** indica che il corpo è un messaggio completo, che comprende intestazione e corpo. Nonostante il nome di questo sottotipo, il messaggio incapsulato potrebbe non essere un comune messaggio RFC 822, ma un qualunque messaggio MIME.

Il sottotipo **message/partial** consente la scomposizione di un messaggio di dimensioni elevate in un certo numero di parti che devono essere riassemblate a destinazione. Per questo sottotipo, nel campo Content-Type: Message/Partial sono specificati tre parametri: un *id* comune a tutti i frammenti dello stesso messaggio, un *numero di sequenza* unico per ciascun frammento e il numero *totale* di frammenti.

Il sottotipo **message/external-body** indica che i dati da trasferire nel messaggio non sono contenuti nel corpo. Al contrario, il corpo contiene le informazioni necessarie ad accedere ai dati. Come per gli altri tipi di messaggio, il sottotipo message/external-body ha un'intestazione esterna e un messaggio incapsulato con la propria intestazione. Il campo Content-Type è l'unico necessario nell'intestazione esterna e identifica il messaggio come avente il sottotipo message/external-body. L'intestazione interna coincide con quella del messaggio incapsulato. Il campo Content-Type nell'intestazione esterna comprende un parametro denominato *access-type* (tipo di accesso), che specifica il metodo di accesso, quale ad esempio FTP (*file transfer protocol*).

Il tipo **application** si riferisce ad altre tipologie di dati, tipicamente dati binari non interpretati o informazioni da elaborare mediante un'applicazione basata su posta elettronica.

Codifiche di trasferimento di MIME

L'altra componente fondamentale della specifica MIME, in aggiunta alla specifica dei tipi di contenuto, è costituita dalla definizione delle codifiche di trasferimento per il corpo dei messaggi. L'obiettivo è quello di fornire una consegna affidabile su un'insieme di ambienti il più grande possibile.

Lo standard MIME definisce due metodi di codifica dei dati. Il campo Content-Transfer-Encoding può assumere sei valori, come riportato nella Tabella 5.4. Tuttavia, tre di questi valori (7bit, 8bit e binary) indicano che non è stata eseguita alcuna codifica, pur fornendo qualche informazione circa la natura dei dati. Per il trasferimento SMTP, è sicuro utilizzare la forma a 7bit. Le forme a 8bit e binaria possono essere utilizzate in altri contesti. Un altro valore del campo Content-Transfer-Encoding è x-token, che sta a indicare l'uso di qualche altro schema di codifica, di cui deve essere fornito il nome. Tale schema può essere quello di uno specifico fornitore o quello di un'applicazione. I due schemi attualmente definiti sono quoted printable e base64. Sono definiti due schemi che forniscono una scelta fra una tecnica di trasferimento essenzialmente leggibile da un utente e una sicura per tutti i tipi di dati, che risulti ragionevolmente compatta.

La codifica di trasferimento **quoted-printable** è utile quando i dati sono composti per lo più da byte corrispondenti a caratteri ASCII stampabili. Essenzialmente, rappresenta caratteri attraverso la rappresentazione esadecimale della loro codifica e introduce interruzioni di riga reversibili (*soft*) per limitare le righe del messaggio a 76 caratteri.

La codifica **base64 transfer**, conosciuta anche come codifica radix-64, è comunemente usata per la codifica di dati binari arbitrari in modo che non possano essere modificati du-

7bit	I dati sono tutti rappresentati da righe brevi di caratteri ASCII.
8bit	Le righe sono brevi, ma ci possono essere caratteri non-ASCII (byte con il bit più alto posto a 1).
binary	Non solo possono essere presenti caratteri non-ASCII ma le righe non sono necessariamente corte a sufficienza per il trasporto SMTP.
quoted-printable	Codifica i dati in modo che se questi sono essenzialmente testo ASCII, la loro forma codificata resta comunque riconoscibile dall'utente.
base64	Codifica i dati trasformando blocchi di 6 bit in ingresso in blocchi di 8 bit in uscita, tutti caratteri ASCII stampabili.
x-token	Codifica non standard con denominazioni.

Tabella 5.4 Codifiche di trasferimento MIME.

rante l'elaborazione da parte dei programmi di trasporto della posta. Tale codifica è anche utilizzata da PGP ed è descritta nell'Appendice 5B.

Un esempio multipart

La Figura 5.8, tratta dalla RFC 2045, delinea un messaggio complesso di tipo multipart. Il messaggio ha cinque parti che devono essere mostrate sequenzialmente: due parti introduttive di testo in formato ASCII, un messaggio contenuto di tipo multipart, una parte con formato richtext e un messaggio di testo di chiusura incapsulato, codificato mediante un insieme di caratteri non-ASCII. Il messaggio multipart contenuto ha due parti, un'immagine e un frammento audio, che devono essere mostrate in parallelo.

Forma canonica

Importante concetto, sia in MIME sia in S/MIME, la forma canonica è un formato, appropriato per il tipo di contenuto, standardizzato per l'uso tra sistemi diversi. Questo concetto è in contrasto con quello della forma nativa, cioè un formato che può essere peculiare di uno specifico sistema. La Tabella 5.5, tratta dalla RFC 2049, dovrebbe essere di aiuto per chiarire meglio questi concetti.

Funzionalità di S/MIME

Rispetto alla funzionalità generale, S/MIME è molto simile a PGP. Entrambi offrono la capacità di firmare e/o cifrare i messaggi. In questo paragrafo sono brevemente sintetizzate le funzionalità di S/MIME, che saranno in seguito esaminate in maggior dettaglio, considerando i formati dei messaggi e la loro preparazione.

Funzionalità

S/MIME fornisce le seguenti funzionalità.

- **Dati con involucro:** questa funzione consiste nella cifratura di qualunque tipo di contenuto e nelle relative chiavi di cifratura per uno o più destinatari.

MIME-Version: 1.0
 From: Nathaniel Borenstein <nsb@hellcore.com>
 To: Ned Freed <ned@innosoft.com>
 Subject: Esempio multipart
 Content-Type: multipart/mixed;
 boundary=unique-boundary-1

Questa è l'area di preambolo di un messaggio multipart. I lettori in grado di comprendere il formato multipart possono ignorare questo preambolo. Chi sta leggendo questo testo, potrebbe considerare di scegliere un programma per la lettura della posta che sia in grado di visualizzare opportunamente i messaggi multipart.

--unique-boundary-1

...Qui compare un testo...

[Si noti che la riga vuota precedente significa che non sono stati forniti campi di intestazione e che questo è testo, con caratteri US ASCII. Avrebbe potuto essere stato creato mediante digitazione esplicita come nella parte successiva.]

--unique-boundary-1

Content-Type: text/plain; charset=US-ASCII

Questa avrebbe potuto essere parte della precedente, ma illustra la digitazione esplicita rispetto a quella non esplicita delle parti del corpo

--unique-boundary-1

Content-Type: multipart/parallel; boundary=unique-boundary-2

--unique-boundary-2

Content-Type: audio/basic

Content-Transfer-Encoding: base64

...dati audio codificati base64 8000 Hz codifica mono-canale vanno qui...

--unique-boundary-2

Content-Type: image/jpeg

Content-Transfer-Encoding: base64

...dati immagine codificati base64 vanno qui...

--unique-boundary-2--

--unique-boundary-1

Content-Type: text/enriched

Questo è <i>richtext.</i> <smaller>come definito nella RFC 1896</smaller>

Non è <bigger><bigger>troppe grandi?</bigger></bigger>

--unique-boundary-1

Content-Type: message/rfc822

From: (mailbox in US-ASCII)

To: (address in US-ASCII)

Subject: (subject in US-ASCII)

Content-Type: Text/plain; charset=ISO-8859-1

Content-Transfer-Encoding: Quoted-printable

...Ulteriore testo in ISO-8859-1 va qui...

--unique-boundary-1--

Figura 5.8 Esempio di struttura di un messaggio MIME.

Forma nativa	Il corpo da trasmettere viene creato nel formato nativo del sistema. Si utilizza l'insieme di caratteri nativo e, dove appropriato, si usano le convenzioni di fine-riga proprie del sistema. Il corpo può essere: un file di testo nello stile UNIX, un'immagine Sun raster, un file VMS indicizzato, dati audio in un formato proprietario del sistema, memorizzato solo in memoria oppure qualunque altro contenuto che corrisponde al modello locale di rappresentazione, per alcuni tipi di informazione. Fondamentalmente, i dati sono creati nel formato nativo corrispondente al tipo specificato dalla tipologia di strumento.
Forma canonica	L'intero corpo, comprese le informazioni al di fuori del corpo stesso come, ad esempio, le lunghezze dei record ed eventuali informazioni su attributi di file, viene convertito in forma canonica universale. La natura della forma canonica usata è dettata dalla specifica tipologia di strumento di supporto del corpo e dai suoi attributi. La conversione alla forma canonica appropriata può richiedere la conversione dell'insieme dei caratteri, la trasformazione dei dati audio, la compressione, o varie altre operazioni che dipendono dalle diverse tipologie di strumenti di supporto. Se si esegue anche una conversione dell'insieme dei caratteri, occorre prestare attenzione alla comprensione del significato della tipologia di strumento di supporto, che potrebbe avere forti implicazioni per qualunque conversione di insiemi di caratteri (ad esempio, con riferimento ai caratteri sintatticamente significativi in un sottotipo testo diverso da "plain").

Tabella 5.5 Forma canonica e forma nativa.

- **Dati firmati:** si costruisce una firma digitale prendendo il digest di messaggio del contenuto da firmare e cifrandolo con la chiave privata del firmatario. Il contenuto e la firma sono quindi cifrati con la codifica base64. Un messaggio firmato può essere visionato solamente da un ricevente con funzionalità S/MIME.
- **Dati in chiaro firmati:** come nel caso precedente, si costruisce una firma digitale del contenuto. Qui, però, si codifica con base64 solo la firma. Come risultato, un ricevente senza funzionalità S/MIME può visionare il contenuto del messaggio, anche se non può verificarne la firma.
- **Dati firmati e con involucro:** entità solo firmate e solo cifrate possono essere nidificate, in modo che dati cifrati possano essere firmati e dati firmati o dati in chiaro firmati possano essere cifrati.

Algoritmi di crittografia

La Tabella 5.6 sintetizza gli algoritmi di crittografia usati in S/MIME. Per specificare la tipologia dei requisiti, S/MIME usa la seguente terminologia, tratta dalla RFC 2119.

- **Must:** la definizione è un requisito assoluto della specifica. Un'implementazione deve comprendere questa caratteristica o funzionalità al fine di essere conforme alla specifica.
- **Should:** in circostanze particolari possono esserci valide ragioni per trascurare questa caratteristica o funzionalità; si raccomanda però che sia contenuta dalle implementazioni.

S/MIME incorpora tre algoritmi a chiave pubblica. L'algoritmo di firma digitale standard (DSS), definito da NIST, è quello preferito per la firma digitale. S/MIME cita Diffie-

Funzionalità	Requisito
Creare un digest di messaggio da usare nella generazione della firma digitale.	Deve (MUST) supportare SHA-1 e MD5. Il ricevitore dovrebbe (SHOULD) supportare MD5 per compatibilità all'indietro.
Cifrare il digest di messaggio per generare la firma digitale.	Gli agenti mittenti e riceventi devono (MUST) supportare DSS. L'agente mittente dovrebbe (SHOULD) supportare la cifratura RSA. L'agente ricevente dovrebbe (SHOULD) supportare la verifica di firme RSA con dimensioni di chiave da 512 a 1024 bit.
Cifratura della chiave di sessione per la trasmissione con il messaggio.	Gli agenti mittenti e riceventi dovrebbero (SHOULD) supportare Diffie-Hellman. Gli agenti mittenti e riceventi devono (MUST) supportare la cifratura RSA con dimensione della chiave da 512 a 1024 bit.
Cifratura del messaggio per la trasmissione con chiave di sessione one-time	Gli agenti mittenti e riceventi devono (MUST) supportare la cifratura con triplo DES. Gli agenti mittenti dovrebbero (SHOULD) supportare la cifratura con AES. Gli agenti mittenti dovrebbero (SHOULD) supportare la cifratura con RC2/40.
Creare un codice di autenticazione del messaggio.	Gli agenti riceventi devono (MUST) supportare HMAC con SHA-1. Gli agenti riceventi dovrebbero (SHOULD) supportare HMAC con SHA-1.

Tabella 5.6 Algoritmi di cifratura usati in S/MIME.

Hellman come algoritmo preferito per la cifratura delle chiavi di sessione; in realtà, S/MIME usa la variante ElGamal di Diffie-Hellman che fornisce cifratura/decifratura. In alternativa, si può usare RSA (Capitolo 3) sia per scopi di firma sia per scopi di cifratura delle chiavi di sessione. Questi algoritmi sono gli stessi usati in PGP e forniscono un livello di sicurezza elevato.

Per quanto riguarda la funzione hash usata per la creazione della firma digitale, la specifica raccomanda SHA-1 a 160 bit ma richiede che il ricevente supporti MD5 a 128 bit, per la compatibilità all'indietro (retrocompatibilità) con le vecchie versioni di S/MIME.

Come detto nel Capitolo 3, c'è preoccupazione giustificabile circa la sicurezza di MD5, quindi SHA-1 è chiaramente l'alternativa preferita.

Per la cifratura dei messaggi, è raccomandato il triplo DES a tre chiavi, ma le implementazioni compatibili devono supportare anche RC2 a 40 bit. Quest'ultimo è un algoritmo di cifratura debole che però garantisce compatibilità con i controlli statunitensi sull'esportazione.

Nella specifica S/MIME è presente una discussione sulla procedura per decidere quale algoritmo utilizzare per la cifratura del contenuto. In sintesi, un agente mittente deve pren-

dere due decisioni. Prima di tutto, deve determinare se l'agente ricevente è in grado di effettuare la decifratura con un determinato algoritmo. Successivamente, se l'agente ricevente è in grado di accettare solamente contenuti con algoritmi di cifratura cifrati deboli, l'agente mittente deve decidere se è accettabile eseguire l'invio effettuando la cifratura mediante tali algoritmi. Per supportare questo processo decisionale, un agente mittente può annunciare le sue capacità di decifratura in ordine di preferenza per ciascun messaggio che invia, e un agente ricevente può memorizzare tale informazione per usi futuri.

Un agente mittente dovrebbe seguire le regole qui di seguito riportate, nell'ordine con cui sono presentate.

- Se l'agente mittente ha una lista delle capacità di decifratura preferite del ricevente designato, dovrebbe (SHOULD) scegliere la prima della lista (quella con preferenza più elevata) tra quelle che è in grado di utilizzare.
- Se l'agente mittente non possiede tale lista, ma ha ricevuto uno o più messaggi da parte di quel ricevente, allora il messaggio in uscita dovrebbe (SHOULD) usare lo stesso algoritmo di cifratura impiegato per l'ultimo messaggio firmato e cifrato pervenuto da quel ricevente.
- Se l'agente mittente non ha conoscenza delle capacità di decifratura del ricevente designato, e decide di rischiare che il ricevente non sia in grado di decifrare il messaggio, allora dovrebbe (SHOULD) usare il triplo DES.
- Se l'agente mittente non ha conoscenza delle capacità di decifratura del ricevente designato, e non desidera rischiare che il ricevente non sia in grado di decifrare il messaggio, allora deve (MUST) usare RC2/40.

Se occorre inviare un messaggio a più destinatari e non è possibile scegliere un algoritmo comune di cifratura per tutti, allora l'agente mittente dovrà inviare due messaggi. Tuttavia, in questo caso, è importante notare che la sicurezza del messaggio è a rischio per il fatto di aver spedito una copia del messaggio utilizzando algoritmi con minori garanzie di sicurezza.

Messaggi S/MIME

S/MIME utilizza un certo numero di nuovi tipi di contenuto MIME, riportati nella Tabella 5.7. Tutti i nuovi tipi applicativi fanno uso della denominazione PKCS (*public-key cryptography specification*). Questa si riferisce a un insieme di specifiche di crittografia a chiave pubblica emesse dagli RSA Laboratories e rese disponibili per S/MIME.

Prima di esaminare ciascun tipo, si considerano le procedure generali di preparazione dei messaggi S/MIME.

Rendere sicura un'entità MIME

S/MIME rende sicura un'entità MIME o mediante firma o mediante cifratura oppure utilizzando entrambe. Un'entità MIME può essere un messaggio completo (ad eccezione delle intestazioni RFC 822); altrimenti, se il tipo di contenuto MIME è multipart, può coincidere con una o più sottoparti del messaggio. L'entità MIME viene approntata secondo le normali regole di preparazione dei messaggi MIME. Quindi, l'entità MIME e alcune informazioni relative alla sicurezza, quali identificatori di algoritmi e certificati, vengono elabo-

Tipo	Sottotipo	Parametro smime	Descrizione
Multipart	Signed		Un messaggio clear-signed in due parti: una è il messaggio e l'altra la firma.
Application	pkcs7-mime	signedData	Un'entità S/MIME firmata.
	pkcs7-mime	envelopedData	Un'entità S/MIME cifrata.
	pkcs7-mime	degenerate signedData	Un'entità contenente solo certificati a chiave pubblica.
	pkcs7-mime	CompressedData	Un'entità S/MIME compressa.
	pkcs7-signature	SignedData	Il tipo di contenuto della sottoparte di firma di un messaggio multipart/signed.

Tabella 5.7 Tipi di contenuti S/MIME.

rate da S/MIME per produrre il cosiddetto oggetto PKCS. Un oggetto PKCS viene poi trattato come il contenuto di un messaggio e racchiuso in MIME (con intestazioni MIME appropriate). Questo processo sarà chiaro quando si prenderanno in considerazione – con esempi – oggetti specifici.

Comunque sia, il messaggio da spedire è convertito in forma canonica. In particolare, dati un certo tipo e sottotipo, si utilizza la forma canonica appropriata per il contenuto del messaggio. Nel caso di messaggi multipart, viene utilizzata la forma canonica appropriata per ciascuna sottoparte.

L'uso della codifica di trasferimento necessita di particolare attenzione. Nella maggior parte dei casi, come risultato dell'applicazione dell'algoritmo di sicurezza, viene generato un oggetto parzialmente o totalmente rappresentato sotto forma di dati binari arbitrari. Questo viene successivamente racchiuso in un altro messaggio MIME, e a questo punto è possibile applicare la codifica di trasferimento, tipicamente base64. Comunque, nel caso di un messaggio multipart firmato – successivamente descritto in maggior dettaglio – il contenuto del messaggio in una delle sottoparti è lasciato inalterato dalla procedura di sicurezza. A meno che quel contenuto non sia di 7 bit, dovrebbe essere codificato per il trasferimento con base64 o quoted printable, in modo da non rischiare di modificare il contenuto cui è stata applicata la firma.

Vengono ora presi in esame i diversi tipi di contenuto S/MIME.

EnvelopedData

Si utilizza un sottotipo application/pkcs7-mime per una delle quattro categorie di elaborazione S/MIME, ciascuna con un parametro smime-type univoco. In ogni caso, l'entità risultante, denominata *oggetto*, è rappresentata in una forma conosciuta come BER (*basic encoding rules*), definita nel documento ITU-T Recommendation X.209. Il formato BER consiste in stringhe arbitrarie di otto byte e quindi si tratta di dati binari. Questo oggetto dovrebbe essere codificato per il trasferimento tramite base64 nel messaggio MIME esterno. Consideriamo dapprima il tipo di contenuto envelopedData.

I passi per la preparazione di un'entità MIME envelopedData sono i seguenti.

1. Generare una chiave di sessione pseudocasuale per un particolare algoritmo di crittografia simmetrica (RC2/40 o triplo DES).
2. Per ciascun ricevente, cifrare la chiave di sessione mediante la chiave pubblica RSA del ricevente stesso.
3. Per ciascun ricevente, preparare un blocco, denominato RecipientInfo, contenente un identificativo del certificato a chiave pubblica del ricevente³, un identificatore dell'algoritmo utilizzato per la crittografia della chiave di sessione e la chiave di sessione cifrata.
4. Cifrare il contenuto del messaggio con la chiave di sessione.

I blocchi RecipientInfo seguiti dal contenuto cifrato costituiscono l'envelopedData. Questa informazione è quindi codificata in formato base64. Ecco un esempio di messaggio di questo tipo (senza le intestazioni RFC 822).

```
Content-Type: application/pkcs7-mime; smime-type=envelopeddata;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
rFvbnj756tbBghyHhHUujhjhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhjh4VQpfyF467GhIGfHfYGTrfvbnjT6jh7756tbB9H
f8HHGTrfvjhjh776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
OGhIGfHfObnj756yt64v
```

Per ricostruire il messaggio in chiaro, il ricevente prima di tutto toglie la codifica base64; quindi utilizza la propria chiave privata per ricostruire la chiave di sessione e, infine, il contenuto del messaggio viene decifrato mediante la chiave di sessione stessa.

SignedData

Di fatto, il tipo signedData può essere utilizzato con uno o più firmatari. Per chiarezza, ci si limita a descrivere il caso di un'unica firma digitale. I passi per la preparazione di un'entità MIME signedData sono i seguenti.

1. Selezionare un algoritmo per la generazione del digest di messaggio (SHA o MD5).
2. Calcolare il digest di messaggio o la funzione hash del contenuto da firmare.
3. Cifrare il digest di messaggio con la chiave privata del firmatario.
4. Preparare un blocco, denominato SignerInfo, contenente: il certificato a chiave pubblica del firmatario, un identificatore dell'algoritmo utilizzato per la creazione del digest di messaggio, un identificatore dell'algoritmo utilizzato per la crittografia del digest di messaggio e il digest di messaggio cifrato.

L'entità signedData consiste in una serie di blocchi che comprende: l'identificatore dell'algoritmo utilizzato per la creazione del digest di messaggio, il messaggio da firmare e SignerInfo. L'entità signedData può anche comprendere un insieme di certificati a chiave pubblica suf-

³ Si tratta di un certificato X.509, descritto più avanti nel paragrafo.

ficienti a costituire una catena a partire da una entità radice riconosciuta o da un'autorità certificativa di massimo livello fino al firmatario. Questa informazione viene quindi codificata nel formato base64. Ecco un esempio (senza le intestazioni RFC 822):

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
  name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
567GhIGfHfYT6ghyHhUujpfyF4f8HHGTrfvhJhjH776tbB9HG4vQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhUujhjhjH
HUujhjh4VQpfyF467GhIGfHfYGTrfvbnjT6jh7756tbB9H7n8HHGghyHh
6YT64v0GhIGfHfQbnj75
```

Per ricostruire il messaggio firmato e verificarne la firma, il ricevente prima di tutto toglie la codifica base64; quindi, utilizza la chiave pubblica del firmatario per decifrare il digest di messaggio e, infine, il ricevente calcola a sua volta il digest di messaggio e lo confronta con quello decifrato, per verificare la firma.

Clear signing

Il contenuto clear signing si ottiene utilizzando il tipo di contenuto multipart con sottotipo signed. Come già menzionato, questo processo di firma non effettua trasformazioni del messaggio da firmare, cosicché il messaggio viene inviato in chiaro. Pertanto, riceventi con funzionalità MIME, ma non S/MIME, sono in grado di leggere il messaggio in arrivo.

Un messaggio multipart/signed è composto da due parti. La prima può essere un qualsiasi tipo MIME ma preparata in modo da non venire modificata durante il trasferimento dalla sorgente alla destinazione. Vale a dire che, se la prima parte non è di 7 bit, allora necessita di essere codificata con base64 o quoted printable. Quindi, la parte viene elaborata nello stesso modo del tipo signedData, ma in questo caso creando un oggetto con formato signedData avente il campo del contenuto vuoto. Questo oggetto rappresenta una firma separata (*detached*). L'oggetto viene poi codificato per il trasferimento utilizzando base64 per diventare la seconda parte del messaggio multipart/signed. Questa seconda parte ha tipo di contenuto application e sottotipo pkcs7-signature. Ecco un esempio di messaggio di questo tipo.

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
---boundary42
Content-Type: text/plain,
Questo è un messaggio clear-signed.
---boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhUujhjh77n8HHGTrfvbnj756tbB9HG4vQpfyF467GhIGfHfYT6
4vQpfyF467GhIGfHfYT6jh77n8HHGghyHhUujhjh756tbB9HGTrfvbnj
n8HHGTrfvjhjh776tbB9HG4vQbnj7567GhIGfHfYT6ghyHhUujpfyF4
```

```
7GhIGfHfYT64vQbnj756
---boundary42---
```

Il parametro protocol indica che si tratta di un'entità in due parti, di tipo clear-signed. Il parametro micalg indica il tipo di digest di messaggio utilizzato. Il ricevente può verificare la firma prendendo il digest di messaggio della prima parte e confrontandolo con il digest di messaggio ricostruito a partire dalla firma nella seconda parte.

Richiesta di registrazione

Tipicamente, un utente o un'applicazione si rivolge a un'autorità certificativa per ottenere certificati a chiave pubblica. L'entità S/MIME application/pkcs10 è utilizzata per trasferire una richiesta di certificazione. Questa comprende un blocco certificationRequestInfo, seguito da un identificatore dell'algoritmo di cifratura a chiave pubblica, seguito a sua volta dalla firma del blocco certificationRequestInfo, eseguita utilizzando la chiave privata del mittente. Il blocco certificationRequestInfo comprende una denominazione del soggetto del certificato (l'entità la cui chiave pubblica deve essere certificata) e una rappresentazione della chiave pubblica dell'utente sotto forma di stringa di bit.

Messaggio certificates-only

In risposta a una richiesta di registrazione, si può inviare un messaggio contenente solamente certificati o una lista di revoca di certificati (CRL, *certificate revocation list*). Il messaggio ha tipo/sottotipo application/pkcs7-mime con parametro smime degenerate. I passi coinvolti sono gli stessi di quelli necessari alla creazione di un messaggio signedData, ad eccezione del fatto che non c'è contenuto del messaggio e il campo signerInfo è vuoto.

Elaborazione dei certificati in S/MIME

S/MIME utilizza certificati a chiave pubblica conformi alla versione 3 di X.509 (Capitolo 4). Lo schema di gestione delle chiavi usato da S/MIME è in un certo senso un ibrido fra una stretta gerarchia di certificazione X.509 e la rete di fiducia di PGP. Come nel modello PGP, gli amministratori e/o utenti S/MIME devono configurare ciascun client con una lista di chiavi fidate e con una lista di revoca dei certificati. Ovvero, la responsabilità di mantenere i certificati necessari alla verifica delle firme in ingresso e alla cifratura dei messaggi in uscita, è locale. D'altra parte, i certificati sono firmati da autorità certificative.

Ruolo dell'agente utente

Un utente S/MIME ha il compito di eseguire svariate funzioni di gestione delle chiavi.

- **Generazione di chiavi:** l'utente responsabile di funzionalità amministrative connesse (per esempio, una funzionalità di gestione di LAN) deve (MUST) essere in grado di generare coppie di chiavi separate Diffie-Hellman e DSS e dovrebbe (SHOULD) essere in grado di generare coppie di chiavi RSA. Ciascuna coppia di chiavi deve (MUST) essere generata da una buona sorgente di ingressi casuali non deterministici ed essere protetta in maniera sicura. Un agente utente dovrebbe (SHOULD) generare coppie di chiavi RSA con lunghezza compresa nell'intervallo da 768 a 1024 bit e non deve (MUST NOT) generare chiavi con lunghezza inferiore a 512 bit.

- Registrazione:** la chiave pubblica di un utente deve essere registrata presso un'autorità certificativa al fine di ricevere un certificato a chiave pubblica X.509.
- Memorizzazione e reperimento dei certificati:** un utente ha necessità di accedere a una lista locale di certificati al fine di verificare le firme in arrivo e di cifrare messaggi in uscita. Tale lista potrebbe essere mantenuta dall'utente o da qualche entità amministrativa locale per conto di un certo numero di utenti.

Certificati VeriSign

Numerose aziende forniscono servizi di autorità certificativa (CA). Ad esempio, Nortel ha progettato una soluzione CA d'impresa e può fornire supporto S/MIME nell'ambito di un'organizzazione. Esistono molte CA basate su Internet, fra cui VeriSign, GTE, e U.S. Postal Service. Fra queste, il servizio CA più diffuso è quello fornito da VeriSign, di cui segue una breve descrizione.

VeriSign fornisce un servizio CA concepito per essere compatibile con S/MIME e con molte altre applicazioni. VeriSign emette certificati X.509 denominati VeriSign Digital ID. All'inizio del 1998, oltre 35 mila siti web commerciali usavano certificati VeriSign Server Digital ID, e sono stati emessi oltre un milione di Digital ID consumer per utenti con browser Netscape e Microsoft.

Le informazioni contenute in un Digital ID dipendono dal suo tipo e dal suo utilizzo. Di base, ciascun Digital ID contiene:

- chiave pubblica del proprietario
- nome o alias del proprietario
- data di scadenza del Digital ID
- numero di serie del Digital ID
- nome dell'autorità certificativa che ha emesso il Digital ID
- firma digitale dell'autorità certificativa che ha emesso il Digital ID.

I Digital ID possono anche contenere ulteriori informazioni fornite dall'utente, fra cui:

- indirizzo
- indirizzo di posta elettronica
- informazioni di registrazione essenziali (nazione, cap, età, sesso).

VeriSign fornisce tre livelli, o classi, di sicurezza per i certificati a chiave pubblica, come sintetizzato nella Tabella 5.8. Un utente richiede in linea un certificato al sito web di VeriSign o ad altri siti web partecipanti. Le richieste di Classe 1 e Classe 2 sono elaborate in linea, e nella maggior parte dei casi richiedono pochi secondi per essere approvate. Ecco le procedure utilizzate.

- Per Digital ID di Classe 1, VeriSign conferma l'indirizzo di posta elettronica dell'utente inviando un codice segreto (PIN) e le informazioni per prelevare il Digital ID all'indirizzo di posta elettronica fornito nell'applicazione.

	Sintesi della conferma dell'identità	Protezione delle chiavi private della IA	Protezione dei certificati e delle chiavi private	Applicazioni realizzate o utilizzate dagli utenti
Classe 1	Ricerca automatica del nome e dell'indirizzo di posta elettronica senza ambiguità	PCA; hardware fidato; CA; software fidato o hardware fidato	Software di cifratura (progetto da PIN) raccomandato ma non richiesto	Browser web e un certo uso della posta elettronica
Classe 2	Come la Classe 1, con in più verifica automatica dell'informazione di iscrizione e dell'indirizzo	PCA e CA; hardware fidato	Software di cifratura (progetto da PIN) richiesto del software	Posta elettronica individuale e intra e inter-aziendale, sottoscrizione on-line, sostituzione delle password e validazione
Classe 3	Come la Classe 1, con in più verifica di persona e documenti di identificazione, verifica automatica dell'identità degli individui della Classe 2; registrazioni aziendali (o archiviazioni su file) per le organizzazioni	PCA e CA; hardware fidato	Software di cifratura (progetto da PIN) richiesto; token hardware raccomandato ma non richiesto	E-banking, accesso a dati aziendali, applicazioni banking Personale, servizi in linea Per i membri servizi di integrità dei contenuti, server di commercio elettronico, validazione del software/autenticazione di LRAA; cifratura forte per alcuni server

IA: autonoma emittente (issuing authority)
 CA: autorità certificativa (certification authority)
 PCA: autorità certificativa pubblica primaria VeriSign (VeriSign public primary certification authority)
 PIN: codice di identificazione personale (personal identification number)
 LRAA: amministratore dell'autorità di registrazione locale (local registration authority administrator)

Tabella 5.8 Classi di certificati a chiave pubblica VeriSign.

- Per Digital ID di Classe 2, VeriSign verifica le informazioni nell'applicazione mediante confronto automatico con una base di dati dei consumatori, oltre a eseguire tutti i controlli previsti per i Digital ID di Classe 1. Infine, viene inviata una conferma all'indirizzo postale specificato, avvisando l'utente che è stato emesso a suo nome un Digital ID.
- Per Digital ID di Classe 3, VeriSign richiede un livello superiore di certificazione dell'identità. Un individuo deve provare la propria identità fornendo credenziali con attestazione di autenticità (*notarization*) oppure richiedendo il certificato di persona.

Servizi di sicurezza potenziati

Al momento in cui il libro è stato scritto, sono stati proposti tre servizi di sicurezza potenziati come documenti Internet draft. I loro dettagli possono cambiare e ulteriori servizi potrebbero essere aggiunti. I tre servizi sono i seguenti.

- **Ricevute firmate.** Una ricevuta firmata può essere richiesta all'interno di un oggetto SignedData. Restituire una ricevuta firmata fornisce al creatore del messaggio una prova della consegna, per dimostrare a una terza parte che il ricevente ha effettivamente ricevuto il messaggio. In sostanza, il ricevente firma l'intero messaggio originario, compresa la firma del mittente, e aggiunge la nuova firma per formare un nuovo messaggio S/MIME.
- **Etichette di sicurezza.** Si può includere un'etichetta di sicurezza negli attributi autenticati di un oggetto SignedData. Un'etichetta di sicurezza è un insieme di informazioni di sicurezza relative al livello di sensibilità del contenuto protetto dall'incapsulazione S/MIME. Le etichette possono essere utilizzate per il controllo dell'accesso, specificando quali utenti sono abilitati ad accedere a un oggetto. Altri possibili impieghi delle etichette di sicurezza comprendono priorità (segreto, riservato, ristretto e così via) oppure ruoli, tramite la descrizione delle tipologie di persone che possono accedere alle informazioni (per esempio, équipe medica di un paziente, personale paramedico).
- **Mailing list sicure.** Quando un utente invia un messaggio a più destinatari, è necessaria una specifica elaborazione per ciascun ricevente, compreso l'uso della chiave pubblica del ricevente. L'utente può essere esentato da questo compito impiegando i servizi di un agente mail list (MLA, *mail list agent*) di S/MIME. Un MLA può ricevere un singolo messaggio, eseguire la cifratura specifica del ricevente per ciascuno dei riceventi e inoltrare il messaggio. Il creatore del messaggio deve solo inviare il messaggio a MLA, con cifratura eseguita utilizzando la chiave pubblica di MLA.

Siti web consigliati

- **PGP Home Page:** sito web PGP di Network Associates, il più importante venditore commerciale PGP.
- **Home page di International PGP:** progettata per promuovere l'uso su scala mondiale di PGP. Contiene documenti e collegamenti interessanti.
- **Sito MIT di distribuzione di PGP:** il maggiore distributore di PGP freeware. Contiene FAQ, altre informazioni e collegamenti ad altri siti PGP.

- **PGP Charter:** le ultime RFC e gli ultimi draft Internet della Specifica aperta di PGP.
- **S/MIME Charter:** la più recente documentazione RFC e Internet draft per S/MIME.

5.3 Domande di revisione ed esercizi

Domande di revisione

- 5.1 Quali sono i cinque servizi principali forniti da PGP?
- 5.2 Qual è l'utilità di una firma separata?
- 5.3 Perché PGP genera una firma prima di applicare la compressione?
- 5.4 Che cos'è la conversione R64?
- 5.5 Perché la conversione R64 è utile per un'applicazione di posta elettronica?
- 5.6 Perché le funzioni di segmentazione e riassemblaggio sono necessarie in PGP?
- 5.7 Come PGP usa il concetto di livello di fiducia?
- 5.8 Che cos'è RFC 822?
- 5.9 Che cos'è MIME?
- 5.10 Che cos'è S/MIME?

Esercizi

- 5.1 PGP fa uso della modalità cipher feedback (CFB) di CAST-128, mentre la maggior parte delle applicazioni di cifratura simmetrica (diverse dalla cifratura delle chiavi) usano la modalità *cipher block chaining* (CBC). Abbiamo:

$$\text{CBC: } C_i = E(K, [C_{i-1} \oplus P_i]); \quad P_i = C_{i-1} \oplus D(K, C_i)$$

$$\text{CFB: } C_i = P_i \oplus E(K, C_{i-1}); \quad P_i = C_i \oplus E(K, C_{i-1})$$
 Entrambe sembrano fornire lo stesso livello di sicurezza. Suggerite una ragione per cui PGP utilizza la modalità CFB.
- 5.2 Nello schema PGP, qual è il numero atteso di chiavi di sessione generate prima che venga prodotta una chiave che è già stata precedentemente creata?
- 5.3 In PGP, qual è la probabilità che un utente con N chiavi pubbliche abbia almeno un ID di chiave duplicato?
- 5.4 I primi 16 bit del digest di messaggio di 128 bit in una firma PGP sono convertiti in chiaro.
 - a. Fino a che punto ciò compromette la sicurezza dell'algoritmo hash?
 - b. Fino a che punto ciò esegue la sua funzione attesa, ovvero aiutare a determinare se è stata usata la chiave RSA corretta per decifrare il digest?
- 5.5 Nella Figura 5.4, ciascuna voce nel keyring pubblico contiene un campo di fiducia del proprietario, che indica il grado di fiducia attribuito al proprietario della chiave pubblica in questione. Perché questo non è sufficiente? Vale a dire, se questo proprietario è fidato e si suppone che sia il proprietario della chiave pubblica, perché quel grado di fiducia non è sufficiente a consentire a PGP di utilizzare la chiave pubblica in questione?
- 5.6 Si consideri la conversione radix-64 come una forma di cifratura. In questo caso, non c'è chiave. Si supponga però che un avversario sia a conoscenza soltanto del fatto che, per cifrare il testo,

- è stata applicata una certa forma di algoritmo di sostituzione. Quanto risulta efficace questo algoritmo rispetto alla crittoanalisi?
- 5.7 Phil Zimmermann ha scelto IDEA, triplo DES a tre chiavi e CAST-128 come algoritmi di cifratura simmetrici per PGP. Fornire le motivazioni per cui ciascuno dei seguenti algoritmi di cifratura simmetrici descritti in questo libro è più o meno adatto a PGP: DES, triplo DES a due chiavi e AES.

Appendice 5A Compressione dei dati mediante ZIP

PGP fa uso del software di compressione ZIP, scritto da Jean-lup Gailly, Mark Adler, e Richard Wales. ZIP è un software freeware scritto in C che gira come programma di utilità su UNIX e su altri sistemi. Dal punto di vista funzionale, ZIP è equivalente a PKZIP, un software shareware sviluppato da PKWARE Inc., molto diffuso nei sistemi Windows. L'algoritmo ZIP è forse la tecnica di compressione più comunemente utilizzata su piattaforme diverse; sono disponibili versioni freeware e shareware per Macintosh, UNIX, Windows e altri sistemi.

ZIP e gli algoritmi analoghi derivano dall'attività di ricerca di Jacob Ziv e Abraham Lempel. Nel 1977, i due ricercatori descrissero una tecnica basata su un buffer *sliding-window* (a finestra scorrevole) che contiene il testo più recentemente elaborato [ZIV77]. Questo algoritmo è in genere conosciuto come LZ77. Nello schema di compressione ZIP (PKZIP, gzip, zipit, etc.) si fa uso di una versione di questo algoritmo.

LZ77 e le sue varianti sfruttano la probabilità che parole e frasi in un flusso di testo (sequenze di pixel nel caso di dati GIF) siano ripetute. Quando si verifica una ripetizione, questa può essere sostituita con un codice corto. Il programma di compressione ricerca le ripetizioni e sviluppa dinamicamente il codice per la loro sostituzione. Nel tempo si riusano i codici per catturare nuove sequenze. L'algoritmo deve essere concepito in modo tale che il programma di decompressione sia in grado di dedurre le attuali corrispondenze fra codici e sequenze di dati sorgente.

Prima di esaminare dettagliatamente l'algoritmo LZ77, consideriamo un semplice esempio⁴. Prendiamo la frase:

the brown fox jumped over the brown foxy jumping frog

lunga 53 byte, ovvero 424 bit. L'algoritmo elabora questo testo da sinistra a destra. Inizialmente, ciascun carattere è trasformato in una sequenza di 9 bit composta da un 1 binario seguito dalla rappresentazione ASCII a 8 bit del carattere. Con il procedere dell'elaborazione, l'algoritmo cerca le sequenze ripetute. Quando ne incontra una, continua la scansione fino al termine della ripetizione. In altre parole, ogni volta che si verifica una ripetizione, l'algoritmo vi inserisce il maggior numero di caratteri possibile. Nella frase citata, la prima sequenza che si incontra è *the brown fox*. Questa sequenza viene sostituita da un puntatore alla sequenza precedente e dalla lunghezza della sequenza stessa. In questo caso, la precedente sequenza *the brown fox* si verifica 26 caratteri prima e ha una lunghezza di 13 caratteri. Per questo esempio, si ipotizzano due opzioni di codifica: un puntatore di 8 bit e una lunghezza di 4 bit oppure un puntatore di 12 bit e una lunghezza di 6 bit; un'intestazione di 2 bit indica quale opzione viene scelta, dove 00 indica la prima opzione e 01 la seconda. Pertanto, la seconda occorrenza di *the brown fox* viene codificata come <00_b><26_d><13_d>, oppure 00 00011010 1101.

Le rimanenti parti del messaggio compresso sono la lettera *y*, la sequenza <00_b><27_d><5_d>, che sostituisce la sequenza composta dal carattere di spazio seguito da *jump*, e la sequenza di caratteri *ing frog*.

⁴ Basato su un esempio di [WEIS93].

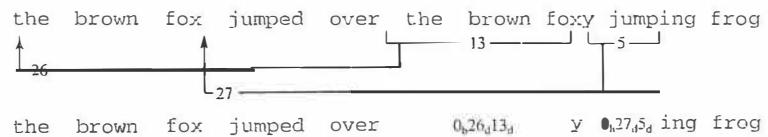


Figura 5.9 Esempio di schema LZ77.

La Figura 5.9 illustra lo schema di compressione. Il messaggio compresso consiste di 35 caratteri a 9 bit e di due codici, per un totale di $35 \times 9 + 2 \times 14 = 343$ bit. Confrontati con i 424 bit del messaggio non compresso, si ottiene un fattore di compressione di 1.24.

Algoritmo di compressione

L'algoritmo di compressione per LZ77 e le sue varianti utilizza due buffer. Un **buffer sliding-history** contiene gli ultimi N caratteri del testo sorgente elaborati, mentre un **buffer look-ahead** contiene i successivi L caratteri da elaborare (Figura 5.10a). L'algoritmo tenta di far combaciare due o più caratteri partendo dall'inizio del buffer look-ahead con la stringa contenuta nel buffer sliding-history. Se non si trovano caratteri combacianti, il primo carattere nel buffer look-ahead viene mandato in uscita sotto forma di carattere a 9 bit e viene anche spostato nella sliding window, rimuovendo il carattere più vecchio ivi contenuto. Se si trovano caratteri combacianti, l'algoritmo continua la scansione per determinare la più lunga stringa di caratteri combacianti. La stringa di caratteri combacianti viene mandata in uscita come una tripla (indicatore, puntatore, lunghezza). Per una stringa di K caratteri, si rimuovono i K caratteri più vecchi contenuti nella sliding window, inserendo al loro posto i K caratteri della stringa codificata.

Il funzionamento di questo schema è mostrato nella Figura 5.10b, con riferimento all'esempio precedentemente introdotto. Nella figura si è ipotizzata una sliding window di 39 caratteri e un buffer look-ahead di 13 caratteri. Nella parte superiore della figura, sono stati elaborati i primi 40 caratteri, mentre la sliding window contiene la versione non compressa degli ultimi 39 caratteri. Il restante testo sorgente è contenuto nella finestra look-ahead. L'algoritmo di compressione determina la suc-

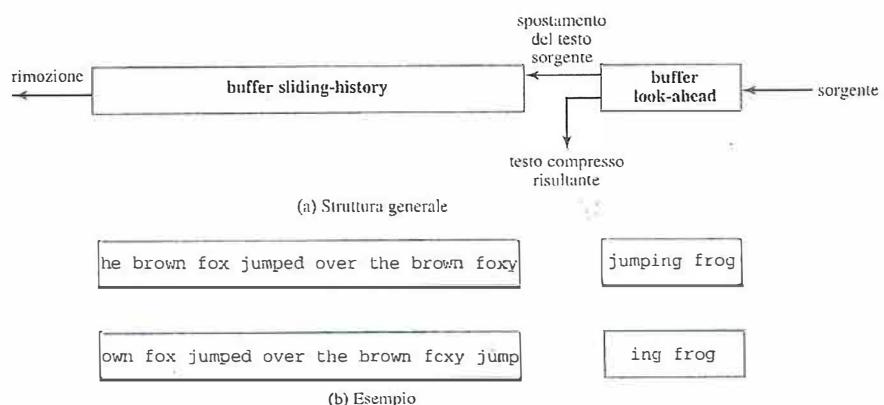


Figura 5.10 Schema LZ77.

siva ripetizione, sposta cinque caratteri dal buffer look-ahead nella sliding window, e produce il codice risultato per questa stringa. La parte sottostante della figura mostra lo stato del buffer dopo queste operazioni.

Pur essendo efficace e capace di adattarsi alla natura dei dati da elaborare, LZ77 presenta alcuni inconvenienti. L'algoritmo utilizza una finestra di dimensione finita per ricercare ripetizioni nel testo. Per blocchi di testo di dimensioni molto elevate, molte potenziali ripetizioni non sono rilevate a causa della dimensione della finestra utilizzata. La dimensione della finestra può essere aumentata, questo comporta però due svantaggi: (1) il tempo di elaborazione dell'algoritmo aumenta in quanto deve eseguire, per ciascuna posizione della sliding window, un confronto di stringhe con il buffer look-ahead e (2) il campo del puntatore deve essere maggiore per adattarsi a salti più lunghi.

Algoritmo di decompressione

La decompressione di testo compresso LZ77 è semplice. L'algoritmo di decompressione deve salvare gli ultimi N caratteri del risultato decompresso. Quando si incontra una stringa codificata, l'algoritmo di decompressione usa i campi <puntatore> e <lunghezza> per sostituire il codice con le corrispondenti stringhe di testo.

Appendice 5B Conversione radix-64

PGP e S/MIME fanno uso di una tecnica di codifica conosciuta come conversione radix-64. Questa tecnica trasforma un ingresso binario arbitrario in un carattere stampabile. La modalità di codifica presenta alcune caratteristiche rilevanti.

- Il dominio della funzione è un insieme di caratteri universalmente rappresentabile in tutti i siti, e non una specifica codifica binaria di quell'insieme di caratteri. Quindi, i caratteri possono essere codificati in qualunque forma richiesta da uno specifico sistema. Ad esempio, il carattere "E" è rappresentato come valore esadecimale 45 in un sistema basato su codifica ASCII e come valore esadecimale C5 in un sistema basato su codifica EBCDIC.
- L'insieme dei caratteri è composto da 65 caratteri stampabili, uno dei quali viene utilizzato per il riempimento. Con $2^6 = 64$ caratteri disponibili, ogni carattere può essere utilizzato per rappresentare 6 bit in ingresso.
- Nell'insieme non sono compresi caratteri di controllo. Quindi, un messaggio codificato in radix-64 può passare attraverso sistemi di gestione di posta che esaminano il flusso dei dati alla ricerca di caratteri di controllo.
- Il carattere *minimo* ("") non è usato. Questo carattere ha un significato nel formato RFC 822 e andrebbe quindi evitato.

La Tabella 5.9 mostra la trasformazione in caratteri dei valori in ingresso di 6 bit. L'insieme dei caratteri è composto dai caratteri alfaniumerici e da "+" e ". Il carattere "=" è usato come riempimento.

La Figura 5.11 illustra il semplice schema di trasformazione. Un ingresso binario viene elaborato a blocchi di 3 byte, o 24 bit. Ciascun insieme di 6 bit nel blocco di 24 bit viene trasformato nel corrispondente carattere. Nella figura si mostrano i caratteri codificati come quantità di 8 bit. In questo specifico caso, ogni ingresso di 24 bit viene espanso a 32 bit nel risultato.

Si consideri, ad esempio, la sequenza di testo di 24 bit 00100011 01011100 10010001, esprimibile in esadecimale come 235C91. Tale ingresso è organizzato in blocchi di 6 bit:

001000 110101 110010 010001

Valore a 6 bit	Carattere codificato						
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(riempimento)	=

Tabella 5.9 Codifica radix-64.

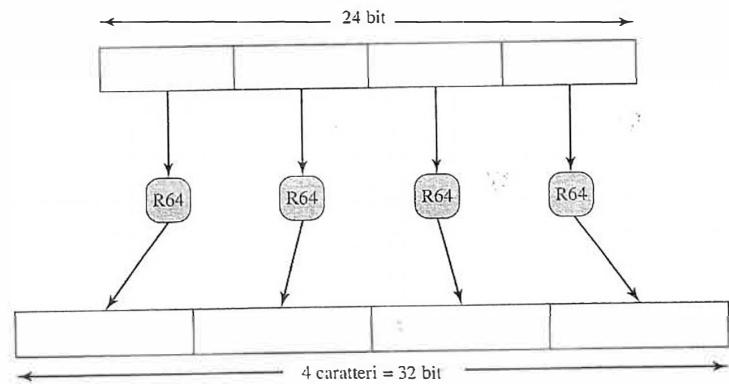


Figura 5.11 Codifica stampabile di dati binari nel formato radix-64.

I valori decimali corrispondenti ai blocchi di 6 bit sono 8, 53, 50, 17. Cercando questi valori nella Tabella 5.9 si genera la seguente codifica radix-64: HyyR. Se questi caratteri sono memorizzati in formato ASCII a 8 bit con bit di parità posto a zero, si ottiene:

01001001 00110001 01111001 01010010

In esadecimale, questo corrisponde a 49317952. In sintesi:

Dati in ingresso	
Rappresentazione binaria	00100011 01011100 10010001
Rappresentazione esadecimale	235C91
Codifica radix-64 dei dati in ingresso	
Rappresentazione a caratteri	HyyR
Codice ASCII (8 bit, parità zero)	01001001 00110001 01111001 01010010
Rappresentazione esadecimale	49317952

Appendice 5C Generazione di numeri casuali in PGP

PGP usa uno schema complesso e potente per la generazione di numeri casuali e pseudocasuali per una serie di scopi. PGP genera numeri casuali partendo dal contenuto e dalla distribuzione temporale delle digitazioni dell'utente, e numeri pseudocasuali utilizzando un algoritmo basato su quello descritto in ANSI X9.17. PGP utilizza le diverse tipologie di numeri per i seguenti scopi:

- numeri casuali veri e propri:
 - usati per generare coppie di chiavi RSA
 - usati come il seme iniziale per il generatore di numeri pseudocasuali
 - usati come un ingresso aggiuntivo durante la generazione di numeri pseudocasuali
- numeri pseudocasuali:
 - usati per generare chiavi di sessione
 - usati per generare vettori di inizializzazione (IV) da utilizzare con chiavi di sessione nella modalità di cifratura CFB.

Numeri casuali veri e propri

PGP mantiene un buffer di 256 byte di bit casuali. Ogni qualvolta PGP aspetta che venga premuto un tasto, registra – in formato a 32 bit – il momento in cui ha avviato la fase di attesa. Quindi, una volta ricevuta la digitazione, memorizza il momento in cui il tasto è stato premuto e il valore a 8 bit della digitazione. L'informazione relativa al momento e al valore della digitazione viene utilizzata per generare una chiave che è a sua volta impiegata per la cifratura del valore corrente del buffer di bit casuali.

Numeri pseudocasuali

La generazione di numeri pseudocasuali utilizza un seme di 24 byte e produce una chiave di sessione di 16 byte, un vettore di inizializzazione di 8 byte e un nuovo seme da impiegare per il successivo ci-

clo di generazione. L'algoritmo utilizza un algoritmo di cifratura simmetrica noto come CAST-128. L'algoritmo usa le seguenti strutture dati.

1. Ingresso.
 - randseed.bin (24 byte): se questo file è vuoto, viene riempito con 24 byte casuali veri e propri.
 - message: la chiave di sessione e IV, che saranno usati per la cifratura del messaggio, sono essi stessi funzione del messaggio. Questo contribuisce ulteriormente alla casualità della chiave e di IV; se un avversario è già a conoscenza del contenuto in chiaro del messaggio, non c'è necessità apparente di acquisire la chiave di sessione one-time.
2. Risultato.
 - K (24 byte): i primi 16 byte, K[0..15], contengono la chiave di sessione, e gli ultimi otto, K[16..23], contengono un IV.
 - randseed.bin (24 byte): in questo file viene posto un nuovo valore di seme.
3. Strutture dati interne.
 - drbuf (8 byte): i primi 4 byte, drbuf[0..3], sono inizializzati con il valore corrente data/tempo. Questo buffer è equivalente alla variabile DT nell'algoritmo X12.17.
 - rkey (16 byte): chiave di cifratura CAST-128 usata in tutte le fasi dell'algoritmo.
 - rseed (8 byte). Equivalente alla variabile V_i di X12.17.
 - rbuf (8 byte): numero pseudocasuale generato dall'algoritmo. Questo buffer è equivalente alla variabile R_i di X12.17.
 - K' (24 byte): buffer temporaneo per il nuovo valore di randseed.bin.

L'algoritmo è articolato in nove passi, da G1 fino a G9. Il primo e l'ultimo sono passi che hanno lo scopo di ridurre l'utilità del file randseed.bin per un avversario che ne sia entrato in possesso. I restanti passi sono mostrati nella Figura 5.12 e sono essenzialmente equivalenti alle tre iterazioni dell'algoritmo X12.17.

- G1. [Pre-elaborazione del seme precedente]
- a. Copiare randseed.bin in K[0..23].
 - b. Considerare il valore hash del messaggio (se il messaggio è stato firmato, tale valore è già stato generato altrimenti si utilizzano i primi 4K byte del messaggio). Usare il risultato come chiave, usare un IV nullo e cifrare K in modalità CFB; memorizzare il risultato nuovamente in K.

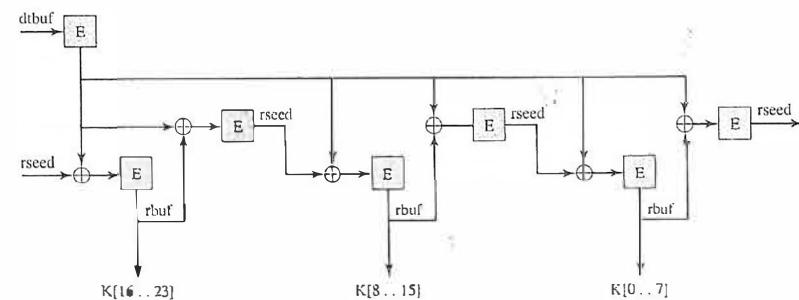


Figura 5.12 Generazione della chiave di sessione e dell'IV in PGP.

```

G2. [Configurazione del seme iniziale]
  a. Porre dtbuf[0..3] al valore a 32 bit del tempo locale. Porre
     dtbuf[4..7] a tutti zero. Copiare rkey ← K[0..15]. Copiare
     rseed ← K[16..23].
  b. Cifrare in modalità ECB il valore a 64 bit di dtbuf utilizzando
     la chiave rkey a 128 bit; memorizzare il risultato nuovamente in dtbuf.

G3. [Preparazione alla generazione dei byte casuali] Porre rcount ← 0 e k ← 23.
  Il ciclo composto dai passi G4–G7 viene eseguito 24 volte (k = 23...0),
  una per ciascun byte casuale prodotto e posto in K. La variabile rcount
  contiene il numero di byte casuali non utilizzati in rbuf; e servirà
  per fare il conto alla rovescia da 8 a 0 per tre volte al fine dei generare
  i 24 byte.

G4. [Test di rcount - I byte sono disponibili?]
  Se rcount=0 goto G5 else goto G7.
  I passi G5 e G6 eseguono un'istanza dell'algoritmo X12.17 per generare
  un nuovo gruppo di otto byte casuali.

G5. [Generazione di nuovi byte casuali]
  a. rseed ← rseed ⊕ dtbuf
  b. rbuf ← Erkey[rseed] in modalità ECB

G6. [Generazione del seme successivo]
  a. rseed ← rbuf ⊕ dtbuf
  b. rseed ← Erkey[rseed] in modalità ECB
  c. Porre rcount ← 8

G7. [Trasferimento byte a byte da rbuf a K]
  a. Porre rcount ← rcount - 1.
  b. Generare un byte casuale vero e proprio b, e porre
     K[k] ← rbuf[rcount] ⊕ b.

G8. [Test sulla fine del ciclo] Se k=0 goto G9 else set k .. k-1 and goto G4.

G9. [Elaborazione finale del seme e generazione del risultato]
  a. Generare 24 ulteriori byte attraverso i passi G4–G7, senza eseguire
     l'OR esclusivo al passo G7. Porre il risultato nel buffer K'.
  b. Cifrare in modalità CFB K' con la chiave K[0..15] e IV K[16..23];
     memorizzare il risultato in randseed.bin.
  c. Restituire K.

```

Non dovrebbe essere possibile determinare la chiave di sessione in base ai 24 nuovi byte generati al passo G9.a. Tuttavia, al fine di garantire che il file randseed.bin memorizzato non fornisca alcuna informazione circa la più recente chiave di sessione, i 24 nuovi byte sono cifrati e il risultato memorizzato come nuovo seme.

Questo complesso algoritmo dovrebbe fornire numeri pseudocasuali robusti dal punto di vista crittografico.

Capitolo 6

Sicurezza IP

La comunità di Internet ha sviluppato meccanismi di sicurezza specifici per parecchie applicazioni, compresa la posta elettronica (S/MIME, PGP) le applicazioni client-server (Kerberos), l'accesso Web e altre. Tuttavia, gli utenti hanno problemi di sicurezza che sono trasversali ai livelli protocolari. Ad esempio, un'azienda può avere una rete privata TCP/IP sicura, non consentendo collegamenti a siti non fidati, cifrando i pacchetti in uscita dalla rete privata e autenticando quelli in entrata.

Un'organizzazione che realizza la sicurezza a livello IP può, quindi, garantire un utilizzo sicuro della rete sia ad applicazioni dotate di meccanismi di sicurezza sia alle numerose applicazioni che non ne dispongono. La sicurezza a livello IP comprende tre aree funzionali: autenticazione, riservatezza e gestione delle chiavi. Il meccanismo di autenticazione assicura, infatti, che ogni pacchetto ricevuto sia stato effettivamente trasmesso dalla parte che nell'intestazione del pacchetto è identificata come sorgente; assicura, inoltre, che il pacchetto non sia stato modificato durante la trasmissione. Il meccanismo che garantisce la riservatezza consente ai nodi in comunicazione di cifrare i messaggi in modo da evitare che vengano intercettati da terze parti. La funzionalità di gestione delle chiavi provvede ad assicurare lo scambio sicuro delle chiavi.

Il presente capitolo inizia con una panoramica sulla sicurezza IP (IPSec, *IP security*) e un'introduzione all'architettura IPSec. Successivamente verranno analizzate in dettaglio le tre aree funzionali precedentemente menzionate. Infine, l'appendice di questo capitolo fornisce una rassegna dei protocolli internet.

6.1 Sicurezza IP: descrizione generale

Per far fronte agli attacchi, l'IAB (*Internet Architecture Board*) ha indicato l'autenticazione e la cifratura come funzionalità di sicurezza indispensabili per il protocollo IP della prossima generazione, protocollo pubblicato con il nome di IPv6. Fortunatamente, tali funzionalità di sicurezza furono progettate per essere utilizzabili sia dall'attuale IPv4 e dal futuro IPv6. Questo significa che i fornitori possono iniziare a offrire queste funzionalità fin da subito, tant'è che molti fornitori hanno già incluso IPSec nei loro prodotti.

Applicazioni di IPSec

IPSec fornisce la possibilità di rendere sicure le comunicazioni su LAN, su WAN private e pubbliche, e su Internet. Ecco alcuni esempi di utilizzo di IPSec.

- Connettività sicura verso delle filiali su Internet.** Una società può costruire una rete privata virtuale sicura su Internet o su una WAN pubblica. Tale possibilità consente di sfruttare massicciamente Internet e riduce quindi la necessità di reti private, consentendo così un risparmio in termini economici e di gestione di rete.
- Accessi remoti sicuri su Internet.** Un utente finale dotato di un sistema in cui siano presenti i protocolli di sicurezza IP può effettuare una chiamata locale a un fornitore di servizi Internet (ISP, *Internet service provider*) e ottenere un accesso sicuro a una rete aziendale. Tale possibilità diminuisce i costi telefonici per gli impiegati che devono effettuare accessi remoti alla rete aziendale.
- Possibilità di stabilire connettività extranet e intranet con i partner.** IPSec può essere utilizzata per rendere sicura la comunicazione con altre organizzazioni, garantendo autenticazione e riservatezza e fornendo un meccanismo per lo scambio di chiavi.
- Miglioramento della sicurezza del commercio elettronico.** Sebbene alcune applicazioni web e di commercio elettronico abbiano propri protocolli di sicurezza, l'utilizzo di IPSec migliora la sicurezza fornita da tali applicazioni.

La caratteristica più rilevante di IPSec è che può cifrare e/o autenticare **tutto** il traffico a livello IP, adattandosi quindi a una notevole gamma di applicazioni. Possono, pertanto, essere rese sicure tutte le applicazioni distribuite, incluse quelle per connessioni remote, applicazioni client/server, posta elettronica, trasferimento di file, accesso al Web, e così via.

La Figura 6.1 illustra un tipico scenario di utilizzo di IPSec. Un'organizzazione dispone di LAN disseminate sul territorio. Il traffico su queste LAN non è sicuro. Per il traffico esterno, effettuato tramite WAN pubblica o privata, sono invece utilizzati i protocolli IPSec. Questi protocolli operano all'interno dei dispositivi di rete, quali ad esempio router o firewall, che connettono ciascuna LAN con l'esterno. I dispositivi di rete IPSec cifrano e comprimono tutto il traffico diretto alla WAN, e decifrano e decomprimono il traffico in arrivo dalla WAN; queste operazioni sono effettuate in modo trasparente rispetto sia alle workstation sia ai server connessi sulle LAN. La trasmissione sicura è possibile anche nel caso di utenti singoli che si collegano alla WAN utilizzando la rete telefonica. Per garantire la sicurezza è necessario che le workstation di tali utenti implementino i protocolli IPSec.

Vantaggi di IPSec

In [MARK97] vengono individuati i seguenti vantaggi derivanti dall'utilizzo di IPSec.

- Quando IPSec è realizzata all'interno di un firewall o di un router, fornisce una sicurezza elevata che può essere applicata a tutto il traffico che attraversa tali dispositivi. Il traffico all'interno di un'organizzazione o di un gruppo di lavoro non subisce rallentamenti a causa delle operazioni da eseguire per garantire la sicurezza IP.
- Quando IPSec viene realizzata all'interno di un firewall non può essere in alcun modo evitata a condizione che tutto il traffico proveniente dall'esterno utilizzi IP, e il firewall sia l'unico mezzo per accedere alla rete dell'organizzazione da Internet.
- IPSec è collocata sotto al livello trasporto (TCP, UDP) e tale caratteristica la rende trasparente rispetto alle applicazioni. Quando IPSec è realizzata in un firewall o in un router non è necessario modificare il software sui sistemi utente o sui server. Anche nel caso

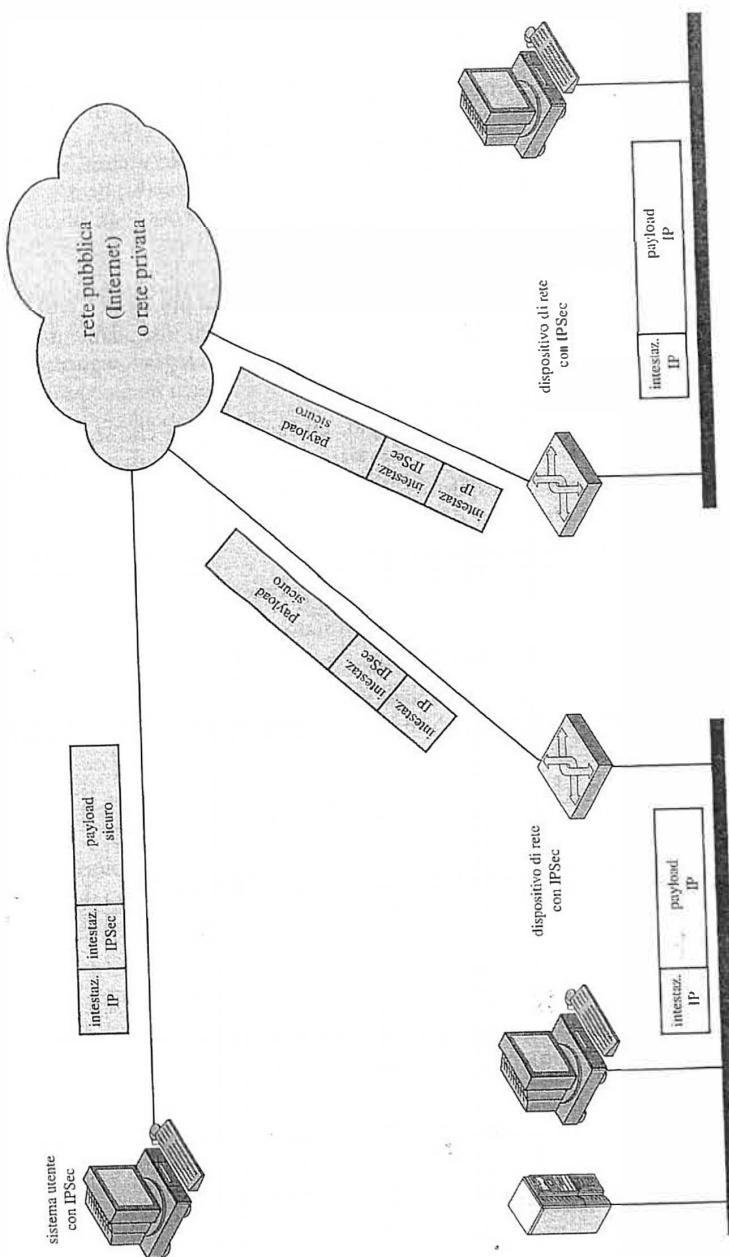


Figura 6.1 Uno scenario di sicurezza IP.

in cui IPSec sia implementata nei sistemi terminali, gli strati software più alti – inclusi le applicazioni – non ne sono influenzati.

- IPSec può essere trasparente all'utente finale. Non c'è alcun bisogno di addestrare gli utenti all'utilizzo dei meccanismi di sicurezza, né di generare le chiavi a livello di singolo utente, o di revocarle quando un utente lascia l'organizzazione.
- Se necessario, IPSec può garantire sicurezza a livello di singolo utente. Tale possibilità è utile nel caso di dipendenti che operano fuori sede o per predisporre, all'interno di un'organizzazione, una sottorete virtuale sicura per applicazioni sensibili.

Applicazioni di instradamento

Oltre al supporto fornito agli utenti finali e alla protezione di sistemi locali e reti, IPSec può giocare un ruolo molto importante all'interno dell'architettura di instradamento necessaria per realizzare ambienti di interconnessione di rete. [HUIT98] elenca i seguenti esempi di utilizzo di IPSec. IPSec può assicurare che:

- un avviso emesso da un router (ad esempio, un nuovo router che segnala la propria presenza) provenga da un router autorizzato;
- un avviso proveniente da un dispositivo vicino (ad esempio, un router che cerca di stabilire o mantenere una relazione di vicinanza con un router in un altro dominio di instradamento) provenga da un router autorizzato;
- un messaggio rediretto provenga dal router cui il pacchetto originario era stato inviato;
- un aggiornamento dell'instradamento non sia stato contraffatto.

Senza tali misure di sicurezza è possibile che un avversario possa interrompere le comunicazioni o deviare una parte del traffico. I protocolli di instradamento – quali, ad esempio, OSPF – dovrebbero essere eseguiti basandosi sulle SA (*security association*) tra router definite da IPSec.

6.2 Architettura di sicurezza IP

Con il passare del tempo, la specifica di IPSec è diventata abbastanza complessa. Per fornire una rappresentazione generale dell'architettura, considereremo innanzi tutto i documenti che definiscono IPSec, tratteremo quindi i servizi IPSec e introdurremo, infine, il concetto di SA (*security association*).

Documenti IPSec

La specifica IPSec è costituita da numerosi documenti: RFC 2401, 2402, 2406 e 2408, pubblicati nel novembre 1998, sono quelli più importanti.

- RFC 2401: fornisce la descrizione generale di un'architettura di sicurezza.
- RFC 2402: contiene la descrizione di un'estensione di IPv4 e IPv6 per l'autenticazione di pacchetti.
- RFC 2406: contiene la descrizione di un'estensione di IPv4 e IPv6 per la cifratura di pacchetti.
- RFC 2408: specifica le funzionalità di gestione delle chiavi.

Le funzionalità sopra elencate devono essere obbligatoriamente fornite da IPv6, mentre sono opzionali nel caso di IPv4. In entrambi i casi, tali funzionalità di sicurezza sono realizzate tramite intestazioni aggiuntive, chiamate intestazioni di estensione, che seguono l'intestazione IP principale. L'intestazione di estensione relativa all'autenticazione è nota con il nome di **intestazione di autenticazione** (AH, *authentication header*) mentre quella per la cifratura è conosciuta come **ESP** (*encapsulating security payload*).

In aggiunta alle quattro RFC sopra descritte, il gruppo di lavoro dell'IETF che si occupa del protocollo di sicurezza IP (*IP Security Protocol Working Group*) ha pubblicato un considerevole numero di bozze aggiuntive. I documenti sono classificati in sette gruppi, illustrati nella Figura 6.2 (RFC 2401).

- **Architettura**: riguarda i concetti generali, i requisiti di sicurezza, le definizioni e i meccanismi alla base della tecnologia IPSec.
- **ESP (encapsulating security payload)**: riguarda il formato dei pacchetti e questioni di carattere generale connesse all'utilizzo di ESP per la cifratura dei pacchetti e, optionalmente, per l'autenticazione.
- **Intestazione di autenticazione** (AH, *authentication header*): copre argomenti riguardanti il formato dei pacchetti e questioni di carattere generale connesse all'utilizzo di AH per l'autenticazione dei pacchetti.

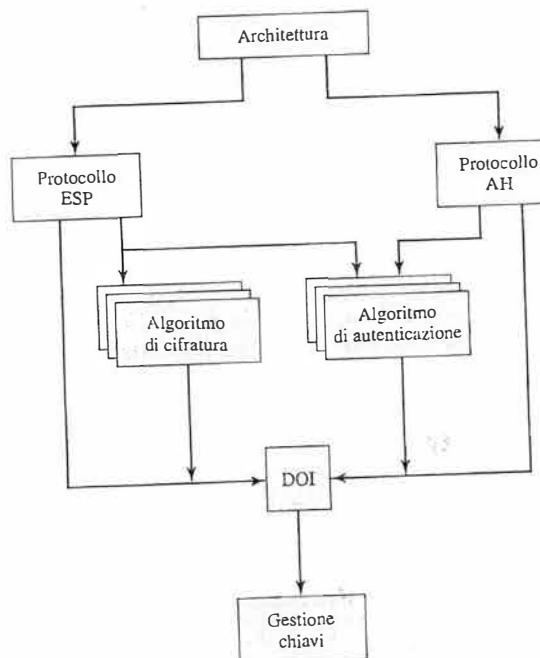


Figura 6.2 Documenti IPSec.

- Algoritmo di cifratura:** insieme di documenti che descrivono come una serie di algoritmi di cifratura sono utilizzati per ESP.
- Algoritmo di autenticazione:** insieme di documenti che descrivono come una serie di algoritmi di autenticazione sono utilizzati per AH e per l'opzione di autenticazione di ESP.
- Gestione delle chiavi:** documenti che descrivono gli schemi di gestione delle chiavi.
- Dominio di interpretazione (DOI, domain of interpretation):** contiene una serie di informazioni necessarie ai documenti delle altre categorie per riferimenti reciproci. Tali informazioni includono gli identificatori di algoritmi di cifratura ed autenticazione che sono stati approvati e parametri operativi quali, ad esempio, il tempo di validità di una chiave.

Servizi IPSec

IPSec fornisce un insieme di servizi di sicurezza a livello IP. Tale funzionalità è ottenuta consentendo a un sistema di selezionare i protocolli di sicurezza che gli occorrono, di decidere gli algoritmi da utilizzare per i servizi e di predisporre tutte le chiavi di cifratura necessarie per fornire i servizi richiesti. La sicurezza è fornita mediante due protocolli: un protocollo di autenticazione, stabilito dall'intestazione del protocollo, cioè dall'intestazione di autenticazione (AH), e un protocollo combinato di cifratura e autenticazione, stabilito dal formato del pacchetto per quello specifico protocollo, cioè dall'ESP. I servizi offerti sono:

- controllo dell'accesso;
- integrità in assenza di connessione;
- autenticazione della sorgente dati;
- rifiuto di pacchetti originati da un attacco di replay (una forma di integrità parziale della sequenza);
- riservatezza (cifratura);
- parziale riservatezza del flusso di traffico.

La Tabella 6.1 mostra quali di questi servizi sono forniti dal protocollo AH e quali dal protocollo ESP. Per quanto riguarda ESP, sono possibili due casi a seconda che sia presente o meno l'opzione di autenticazione. Sia AH che ESP sono strumenti per il controllo degli ac-

	AH	ESP (solo cfrat.)	ESP (cfrat. e autenticaz.)
Controllo dell'accesso	✓	✓	✗
Integrità in assenza di connessione	✓		
Autenticazione della sorgente dati	✓		✓
Rifiuto di pacchetti originali da un attacco di replay	✓		✓
Riservatezza		✓	✓
Parziale riservatezza del flusso di traffico		✓	✓

Tabella 6.1 Servizi IPSec.

cessi, basato sulla distribuzione delle chiavi crittografiche e sulla gestione dei flussi di traffico relativi a quei protocolli di sicurezza.

Associazioni di sicurezza

Un concetto chiave per i meccanismi IP di autenticazione e di riservatezza è l'**associazione di sicurezza** (SA, *security association*). Un'associazione consiste in una connessione logica unidirezionale tra il mittente e il ricevente, che offre servizi di sicurezza al traffico lungo tale connessione. Se è necessaria un'associazione paritetica, al fine di effettuare scambi sicuri in entrambe le direzioni, è necessario utilizzare due associazioni di sicurezza. Nell'ambito di una SA si può utilizzare uno tra i protocolli AH e ESP, ma non entrambi.

Una SA è identificata univocamente da tre parametri.

- Indice dei parametri di sicurezza** (SPI, *security parameters index*): stringa di bit assegnata alla SA in questione con significato a livello locale. SPI è contenuto sia nell'intestazione dell'AH sia in quella dell'ESP, per consentire al sistema ricevente di selezionare la SA preposta alla gestione del pacchetto ricevuto.
- Indirizzo IP destinazione:** allo stato attuale sono permessi solo indirizzi unicast; l'indirizzo destinazione è, quindi, quello dell'entità destinazione della SA. Tale entità può essere sia il sistema di un utente finale sia un dispositivo di rete come, ad esempio, un firewall o un router.
- Identificatore del protocollo di sicurezza:** specifica se si tratta di una SA basata su AH o su ESP.

In ciascun pacchetto IP¹, la SA è quindi univocamente identificata dall'indirizzo destinazione contenuto nell'intestazione IPv4 o IPv6, e da SPI contenuto nell'intestazione di estensione (AH o ESP).

Parametri di una SA

In ogni implementazione IPSec, esiste una base di dati nominale² di SA (SAD, *security association database*) che definisce i parametri associati a ciascuna SA. Normalmente, una SA è definita dai seguenti parametri.

- Contatore del numero di sequenza** (*sequence number counter*): valore di 32 bit utilizzato per generare il campo **numero di sequenza** nelle intestazioni AH o ESP descritte nel Paragrafo 6.3 (parametro obbligatorio per tutte le implementazioni).
- Overflow del contatore di sequenza** (*sequence counter overflow*): flag utilizzato per indicare se un overflow del parametro contatore del numero di sequenza debba essere registrato e debba prevenire ulteriori trasmissioni di pacchetti sulla SA in questione (parametro obbligatorio per tutte le implementazioni).

¹ In questo capitolo, il termine *pacchetto IP* è utilizzato per denotare sia un datagramma IPv4 sia un pacchetto IPv6.

² Nominale nel senso che i servizi offerti da questa base di dati devono essere presenti in ogni implementazione di IPSec, ma il modo di realizzarla costituisce una scelta implementativa.

- **Finestra anti-replay (*anti-replay window*):** utilizzata per determinare se un pacchetto AH o ESP ricevuto oltre il tempo limite costituisce un attacco replay (parametro obbligatorio per tutte le implementazioni). Il servizio anti-replay sarà descritto nel Paragrafo 6.3.
- **Informazioni relative alla AH (*AH information*):** comprendono l'algoritmo di autenticazione, le chiavi, il tempo di validità delle chiavi e ulteriori parametri da utilizzare con AH (parametro obbligatorio per le implementazioni AH).
- **Informazioni relative all'ESP (*ESP information*):** comprendono l'algoritmo utilizzato per la cifratura e l'autenticazione, le chiavi, i valori di inizializzazione, il tempo di validità delle chiavi e ulteriori parametri da utilizzare con ESP (parametro obbligatorio per le implementazioni ESP).
- **Tempo di validità della SA (*lifetime of this security association*):** intervallo di tempo, o numero di byte, dopo il quale una SA deve essere sostituita con una nuova SA (e un nuovo SPI) oppure deve essere terminata. Inoltre, questo parametro indica quale delle due azioni precedentemente descritte deve verificarsi (parametro obbligatorio per tutte le implementazioni).
- **Modalità del protocollo IPSec (*IPSec protocol mode*):** può assumere come valore: tunnel, trasporto o il simbolo di wildcard (parametro obbligatorio per tutte le implementazioni). Le modalità del protocollo IPSec saranno illustrate nel prosieguo del paragrafo.
- **MTU del percorso (*path MTU*):** unità massima di trasmissione (MTU, *maximum transmission unit*) del percorso (è la dimensione massima che un pacchetto può assumere per essere trasmesso senza dover essere frammentato) e variabili che si incrementano col passare del tempo (parametro obbligatorio per tutte le implementazioni).

Il meccanismo di gestione utilizzato per distribuire le chiavi è combinato con i meccanismi di autenticazione e riservatezza solo tramite SPI. Per questa ragione, autenticazione e riservatezza sono state specificate indipendentemente da qualsiasi meccanismo di gestione delle chiavi.

Selettori SA

IPSec fornisce agli utenti una considerevole flessibilità nel modo in cui i servizi IPSec sono applicati al traffico IP. Come si vedrà in seguito, le SA possono essere combinate secondo diverse modalità al fine di produrre la configurazione desiderata dall'utente. Inoltre, IPSec fornisce un elevato livello di granularità per discriminare tra il traffico cui è applicata la protezione IPSec e il traffico cui è consentito aggirare IPSec. In particolare, nel primo caso viene posto in relazione il traffico IP con specifiche SA.

Il mezzo attraverso il quale il traffico IP viene posto in relazione a specifiche SA (o a nessuna SA, nel caso in cui al traffico sia consentito aggirare i controlli IPSec) è la **base di dati delle politiche di sicurezza** (SPD; *security policy database*). Una SPD contiene, nella versione più semplice, una serie di voci, ciascuna delle quali definisce un sottoinsieme del traffico IP e fa riferimento a una SA per quella particolare porzione di traffico. In ambienti più complessi, possono essere presenti più voci che potenzialmente si riferiscono a un'unica SA o a molteplici SA, tutte associate a una singola voce della SPD. Per una trattazione più approfondita, il lettore può consultare i documenti IPSec sull'argomento.

Ogni voce della SPD è definita tramite un insieme di valori di campi del protocollo IP o di protocolli di livelli superiori. Tali valori sono chiamati **selettori**. In ultima analisi, i

selettori sono utilizzati per filtrare il traffico in uscita al fine di metterlo in relazione con una particolare SA. Per ciascun pacchetto IP, vengono effettuati i seguenti passi.

1. Confrontare i valori dei campi appropriati nel pacchetto (i campi selettori) con le informazioni contenute nella SPD al fine di trovare una voce corrispondente, che farà riferimento a zero o più SA.
2. Identificare, se esiste, la SA per il pacchetto in questione e il relativo SPI.
3. Effettuare le elaborazioni richieste da IPSec (cioè le elaborazioni relative a AH o a ESP).

I seguenti selettori determinano una voce SPD.

- **Indirizzo IP destinazione** (*destination IP address*): può essere un singolo indirizzo IP, una lista o un intervallo di indirizzi, oppure un indirizzo contenente wildcard (maschera). Le ultime due modalità sono necessarie nel caso di più sistemi destinazione che condividono la stessa SA (per esempio, protetti dallo stesso firewall).
- **Indirizzo IP sorgente** (*source IP address*): può essere un singolo indirizzo IP, una lista o un intervallo di indirizzi, oppure un indirizzo contenente wildcard. Le ultime due modalità sono necessarie nel caso di più sistemi sorgente che condividono la stessa SA (per esempio, protetti dallo stesso firewall).
- **ID utente** (*userID*): identificatore utente generato dal sistema operativo. Questo campo non è presente nelle intestazioni IP o di livelli superiori ma è disponibile nel caso in cui IPSec sia eseguita nello stesso sistema operativo dell'utente.
- **Livello di sensibilità dei dati** (*data sensitivity level*): utilizzato nel caso di sistemi che forniscono controllo sul flusso delle informazioni (per esempio, "segreto" o "non classificato").
- **Protocollo di livello trasporto** (*transport layer protocol*): ottenuto dal protocollo IPv4 o dal campo **intestazione successiva** di IPv6. Può essere un singolo numero di protocollo, una lista di numeri di protocollo o un intervallo di numeri di protocollo.
- **Porta sorgente e porta destinazione** (*source and destination port*): possono essere singoli valori di porte TCP o UDP, una lista di porte, oppure un'insieme di porte specificate tramite wildcard.

Modalità trasporto e tunnel

Sia AH che ESP possono essere utilizzati secondo due differenti modalità: trasporto e tunnel. Il loro funzionamento sarà meglio compreso nel contesto delle descrizioni di AH ed ESP, rispettivamente fornite nei Paragrafi 6.3 e 6.4. Vediamo ora una breve panoramica delle due modalità.

Modalità trasporto

La modalità trasporto fornisce protezione principalmente per i protocolli dei livelli superiori, vale a dire che la protezione offerta dalla modalità trasporto si estende anche al payload di un pacchetto IP. Esempi includono un segmento TCP o UDP, o un pacchetto ICMP, dal

momento che tutti questi protocolli agiscono immediatamente sopra al protocollo IP nella pila di protocolli di un host. Solitamente, la modalità trasporto è utilizzata per comunicazioni end-to-end tra due host (per esempio, un client ed un server, o due workstation). Quando un host esegue AH o ESP su IPv4, il payload è costituito dai dati che normalmente seguono l'intestazione IP. Nel caso di IPv6, il payload è costituito dai dati che normalmente seguono sia l'intestazione IP sia le intestazioni di estensione di IPv6 presenti, con la possibile eccezione dell'intestazione relativa alle opzioni di destinazione, che può essere inclusa nella protezione.

ESP, in modalità trasporto, cifra e, opzionalmente, autentica il payload IP ma non l'intestazione IP. AH, in modalità trasporto, autentica il payload IP e porzioni selezionate dell'intestazione IP.

Modalità tunnel

La modalità tunnel fornisce protezione all'intero pacchetto IP. A questo scopo, dopo aver aggiunto al pacchetto IP i campi AH o ESP, l'intero pacchetto – con l'aggiunta dei campi di sicurezza – viene considerato come il payload di un nuovo pacchetto IP “esterno” dotato di una sua nuova intestazione IP. L'intero pacchetto originario, o pacchetto interno, viene quindi trasmesso da un punto all'altro di una rete IP, all'interno di un “tunnel”; nessun router lungo il cammino è in grado di esaminare l'intestazione IP interna. Dal momento che il pacchetto originario è incapsulato, il nuovo e più grande pacchetto può avere indirizzi sorgente e destinazione totalmente diversi rispetto a quelli del pacchetto originario, migliorando quindi la sicurezza. La modalità tunnel è utilizzata quando una o entrambe le entità coinvolte in una SA sono gateway di sicurezza (come, ad esempio, un firewall o un router che implementano IPSec). Mediante la modalità tunnel è possibile che un certo numero di host connessi su reti protette da firewall possano realizzare comunicazioni sicure senza la necessità di implementare IPSec. I pacchetti non protetti generati da questi host sono trasmessi in modalità tunnel attraverso reti esterne mediante SA in modalità tunnel, predisposte dal software IPSec posto all'interno del firewall o di un router sicuro, posti ai confini della rete locale.

Ecco un esempio del modo di operare della modalità tunnel di IPSec. L'host A su una certa rete genera un pacchetto IP contenente come indirizzo destinazione quello di un host B connesso a un'altra rete. Tale pacchetto è instradato dall'host che lo ha generato verso un firewall o router sicuro, posto ai confini della rete di A. Il firewall filtra tutti i pacchetti in uscita per determinare se sia necessario attivare le funzionalità IPSec. Se il pacchetto spedito da A a B richiede IPSec, il firewall effettua le necessarie elaborazioni e incapsula il pacchetto in un'intestazione IP esterna. L'indirizzo IP della sorgente di questo pacchetto IP esterno è l'indirizzo del firewall stesso, mentre l'indirizzo destinazione può essere quello di un firewall che delimita il confine della rete locale di B. Il pacchetto è successivamente instradato al firewall di B, e i router intermedi potranno esaminare solo l'intestazione IP esterna. Quando il pacchetto raggiunge il firewall di B, l'intestazione IP esterna viene eliminata, e il pacchetto interno è recapitato a B.

ESP in modalità tunnel cifra e, opzionalmente, autentica l'intero pacchetto IP interno, inclusa la sua intestazione IP. AH in modalità tunnel autentica l'intero pacchetto IP interno e parti selezionate dell'intestazione IP esterna.

La Tabella 6.2 riassume le funzionalità delle modalità trasporto e tunnel.

	SA in modalità trasporto	SA in modalità tunnel
AH	Autentica il payload IP, porzioni selezionate dell'intestazione IP e le intestazioni di estensione IPv6.	Autentica l'intero pacchetto IP interno (intestazione interna e payload IP), porzioni selezionate dell'intestazione IP esterna e le intestazioni di estensione IPv6 esterne.
ESP	Cifra il payload IP e tutte le intestazioni di estensione IPv6 che seguono l'intestazione ESP.	Cifra il pacchetto IP interno.
ESP con autenticazione	Cifra il payload IP e tutte le intestazioni di estensione IPv6 che seguono l'intestazione ESP. Autentica il payload ma non l'intestazione IP.	Cifra il pacchetto IP interno. Autentica il pacchetto IP interno.

Tabella 6.2 Funzionalità delle modalità trasporto e tunnel.

6.3 Intestazione di autenticazione

L'intestazione di autenticazione fornisce un supporto per l'integrità dei dati e l'autenticazione dei pacchetti IP. Il controllo dell'integrità dei dati assicura che non sia possibile effettuare modifiche non rilevate al contenuto di un pacchetto in transito. L'autenticazione consente a un sistema terminale, o a un dispositivo di rete, di autenticare l'utente o l'applicazione, e di filtrare di conseguenza il traffico; previene, inoltre, attacchi volti alla contraffazione di indirizzi (*address spoofing*), oggi abbastanza frequenti su Internet. AH fornisce protezione anche contro gli attacchi di replay, descritti più avanti in questo paragrafo.

L'autenticazione si basa sull'utilizzo di un codice di autenticazione di messaggio (MAC), come descritto nel Capitolo 3; ciò implica che le due parti coinvolte debbano condividere una chiave segreta.

- L'intestazione di autenticazione è composta dai seguenti campi (Figura 6.3).
- **Intestazione successiva, 8 bit (next header):** indica il tipo di intestazione che segue immediatamente l'intestazione in questione.
 - **Lunghezza del payload, 8 bit (payload length):** ottenuta sottraendo due unità alla lunghezza, espressa in parole di 32 bit, di AH. Ad esempio, la lunghezza di default del campo relativo ai dati di autenticazione è 96 bit, ovvero tre parole di 32 bit. Con un'intestazione di lunghezza fissa pari a tre parole, nell'intestazione sono contenute in totale sei parole, e la lunghezza del campo **payload** assume quindi valore 4.
 - **Riservato, 16 bit (reserved):** riservati per utilizzi futuri.
 - **Indice dei parametri di sicurezza, 32 bit (SPI, security parameters index):** identifica una SA.

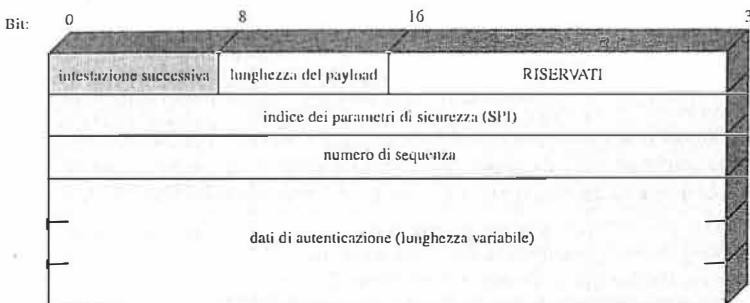


Figura 6.3 Intestazione di autenticazione IPSec.

- Numero di sequenza, 32 bit (*sequence number*):** contatore con valori monotonicamente crescenti, trattato in seguito.
- Dati di autenticazione, lunghezza variabile (*authentication data*):** campo di lunghezza variabile (deve essere un multiplo di parole di 32 bit) contenente il valore per la verifica di integrità (ICV, *integrity check value*), oppure il MAC relativo al pacchetto in questione. Tale campo sarà trattato in seguito.

Servizio anti-replay

Un attacco di replay ha luogo quando un avversario ottiene una copia di un pacchetto autenticato e lo trasmette successivamente alla legittima destinazione. La ricezione di duplicati di pacchetti IP autenticati può provocare malfunzionamenti dei servizi o avere altre conseguenze indesiderate. Il campo numero di sequenza contenuto nell'AH è stato concepito per contrastare questo tipo di attacco. In primo luogo illustreremo come il mittente genera il numero di sequenza e poi descriveremo come viene elaborato dal destinatario.

Quando si costituisce una nuova SA, il **mittente** inizializza a 0 un contatore utilizzato per generare i numeri di sequenza. Ogni volta che viene spedito un pacchetto su questa SA, il mittente incrementa il contatore e inserisce il corrispondente valore nel campo numero di sequenza. Questo significa che il primo valore da utilizzare è 1. Se il servizio anti-replay è abilitato (come avviene di default), il mittente non deve consentire che il valore del numero di sequenza compia un intero ciclo raggiungendo il valore $2^{32} - 1$, ritornando quindi a zero. Se ciò fosse consentito, ci sarebbe più di un pacchetto valido con lo stesso numero di sequenza. Quando viene raggiunto il limite di $2^{32} - 1$, il mittente dovrebbe terminare la SA corrente e negoziare una nuova SA con una nuova chiave.

Dato che IP è un servizio privo di connessione e non affidabile, non garantisce che i pacchetti siano recapitati nello stesso ordine con cui sono stati spediti e nemmeno che tutti i pacchetti raggiungano effettivamente la destinazione. Per questo, il documento IPSec relativo all'autenticazione impone che il **ricevente** realizzzi una finestra di dimensione W , con un valore di default per W pari a 64. Il lato destro della finestra rappresenta il numero di sequenza più alto ricevuto fino a quel momento per un pacchetto valido. Tale numero è denotato con N . Per ogni pacchetto con un numero di sequenza compre-

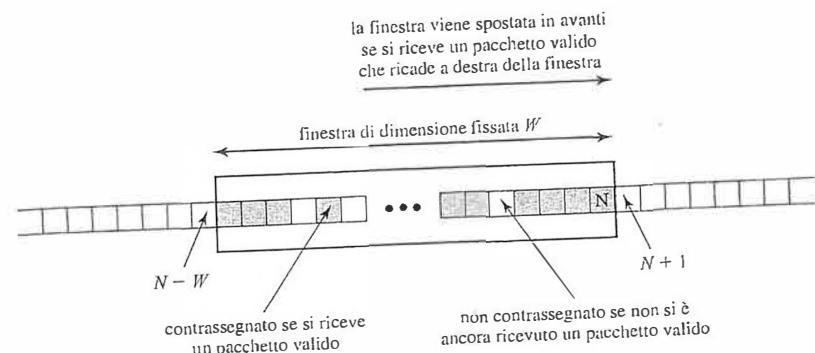


Figura 6.4 Meccanismo anti-replay.

so nell'intervallo da $N - W + 1$ a N ricevuto correttamente (cioè correttamente autenticato), viene contrassegnato il corrispondente slot nella finestra (Figura 6.4). Ogni volta che viene ricevuto un nuovo pacchetto, il processo continua iterativamente, come di seguito illustrato.

- Se il pacchetto ricevuto è nuovo, e ricade all'interno della finestra, viene verificato il MAC. Se il pacchetto è autenticato, viene contrassegnato lo slot corrispondente nella finestra.
- Se il pacchetto ricevuto è nuovo e ricade a destra della finestra, viene verificato il MAC. Se il pacchetto è autenticato, la finestra viene spostata in avanti in modo che il numero di sequenza di tale pacchetto ne costituisca il limite destro, e viene contrassegnato lo slot corrispondente.
- Se il pacchetto ricevuto è alla sinistra della finestra, oppure l'autenticazione non va a buon fine, il pacchetto non viene preso in considerazione; tale azione rappresenta un evento registrabile.

Valore per la verifica di integrità

Il campo **dati di autenticazione** contiene un valore, chiamato valore per la verifica di integrità (ICV). Questo è un MAC o una versione tronca di un codice generato da un algoritmo MAC. La specifica corrente stabilisce che un'implementazione conforme debba fornire i seguenti algoritmi:

- HMAC-MD5-96
- HMAC-SHA-1-96

Entrambi utilizzano l'algoritmo HMAC, il primo con il codice hash MD5 e il secondo con il codice hash SHA-1 (Capitolo 3). In entrambi i casi, è calcolato il valore HMAC completo e, successivamente, viene troncato utilizzando solo i primi 96 bit che rappresentano la lunghezza di default del campo dati di autenticazione.

Il MAC è calcolato considerando i seguenti elementi.

- I campi dell'intestazione IP che non subiscono modifiche durante la trasmissione (immutabili) o il cui valore può essere previsto dalla SA basata su AH al momento del loro arrivo alla destinazione finale. I campi che possono essere modificati durante la trasmissione, o il cui valore all'arrivo non è prevedibile, sono posti a zero per effettuare il calcolo sia alla sorgente sia alla destinazione.
- L'intestazione AH, ad eccezione del campo dati di autenticazione, è posto a zero per effettuare il calcolo sia alla sorgente sia alla destinazione.
- Tutti i dati del protocollo di livello superiore, che si ipotizzano essere immutabili durante la trasmissione (per esempio, un segmento TCP o un pacchetto IP interno in modalità tunnel).

Nel caso di IPv4, sono esempi di campi immutabili la **lunghezza dell'intestazione Internet** (*Internet header length*) e l'**indirizzo sorgente** (*source address*). Un esempio di campo modificabile ma prevedibile è l'**indirizzo destinazione** (*destination address*) con instradamento di sorgente stretto o lasco. Esempi di campi modificabili posti a zero prima di calcolare ICV sono: **tempo di validità** (*time to live*) e **checksum dell'intestazione** (*header checksum*). Si noti che i campi indirizzo sorgente e indirizzo destinazione sono protetti, in modo da prevenirne la contraffazione.

Nel caso di IPv6, un esempio di campo immutabile nell'intestazione base è il campo **versione** (*version*), un esempio di campo modificabile ma prevedibile è l'**indirizzo destinazione**, mentre un esempio di campo modificabile che viene posto a zero prima del computo di ICV è l'**etichetta di flusso** (*flow label*).

Modalità trasporto e tunnel

La Figura 6.5 illustra due modalità di utilizzo del servizio di autenticazione di IPSec. In un caso, l'autenticazione è fornita direttamente tra un server e le workstation client; le workstation possono essere sia sulla stessa rete del server sia su una rete esterna. Il processo di autenticazione è sicuro, a patto che le workstation e il server condividano una chiave segreta

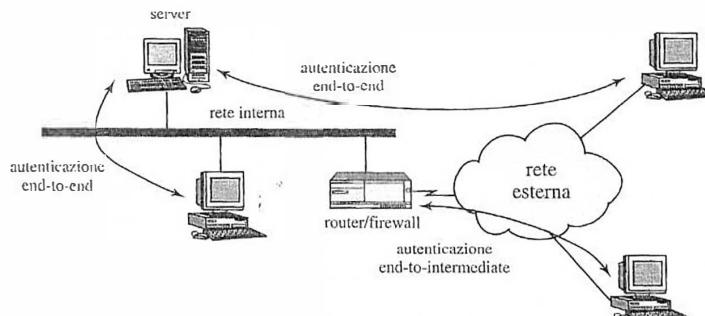


Figura 6.5 Confronto fra l'autenticazione end-to-end e quella end-to-intermediate.

protetta. Il caso illustrato nella figura utilizza una SA in modalità trasporto. Nell'altro caso della figura, una workstation remota si autentica a un firewall aziendale, o per poter accedere all'intera rete interna o perché il server cui si vuole connettere non fornisce meccanismi di autenticazione. In questo caso viene utilizzata una SA in modalità tunnel.

Nel prosieguo della trattazione verranno analizzate, per entrambe le modalità, sia il campo d'azione dell'autenticazione fornita da AH sia il posizionamento dell'intestazione di autenticazione. Le considerazioni da fare sono in alcuni casi diverse per IPv4 e IPv6. La Figura 6.6a illustra la tipica struttura dei pacchetti IPv4 e IPv6. Nel caso considerato, il payload IP è un segmento TCP; potrebbe anche essere un'unità di dati relativa a un qualsiasi altro protocollo che utilizza IP come, ad esempio, UDP o ICMP.

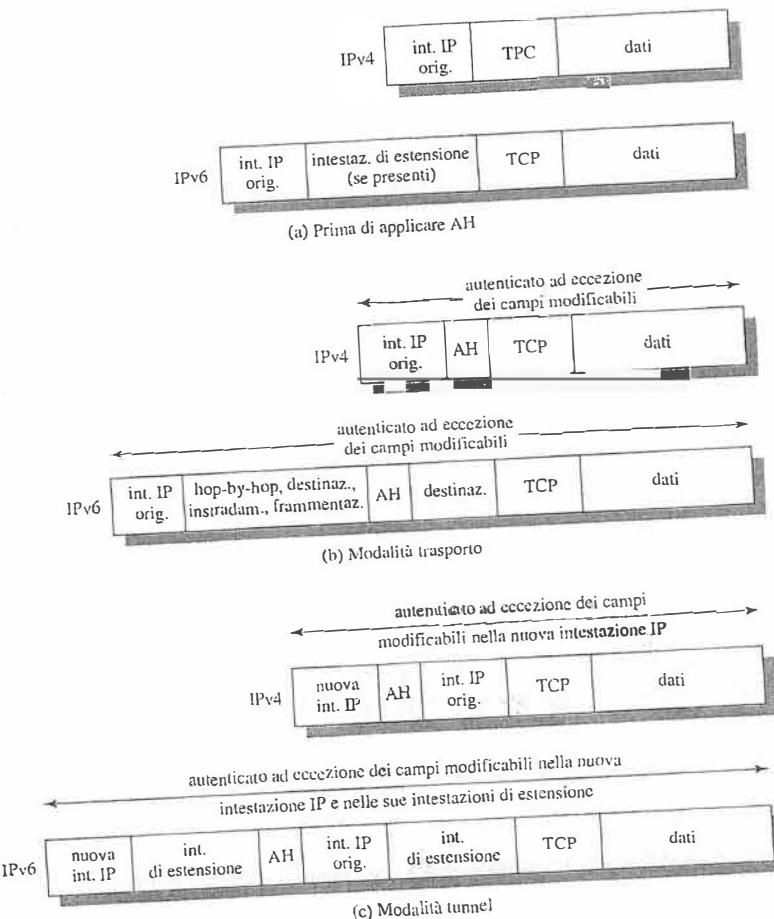


Figura 6.6 Ambito dell'autenticazione fornita da AH.

Nel caso di **AH in modalità trasporto**, se si utilizza IPv4, AH è inserita tra l'intestazione IP originaria e il payload IP (per esempio, un segmento TCP), come illustrato nella parte superiore della Figura 6.6b. L'autenticazione copre l'intero pacchetto, ad eccezione dei campi modificabili nell'intestazione IPv4 che sono posti a zero per il calcolo del MAC.

Nel caso di IPv6, AH è vista come un payload end-to-end; questo significa che non viene esaminata ed elaborata dai router intermedi. Per tale ragione, AH è posta dopo l'intestazione base IPv6 e dopo le intestazioni di estensione relative all'hop-by-hop, all'indirizzamento ed alla frammentazione. L'intestazione di estensione con opzioni per la destinazione può essere posta sia prima sia dopo AH, in dipendenza dal significato che si vuole attribuire. Anche in questo caso, l'autenticazione copre l'intero pacchetto, ad eccezione dei campi modificabili che sono posti a zero per il calcolo del MAC.

Nel caso di **AH in modalità tunnel** viene autenticato l'intero pacchetto IP originario e AH è collocata tra l'intestazione IP originaria e una nuova intestazione IP esterna (Figura 6.6c). L'intestazione IP interna contiene gli indirizzi della sorgente e della destinazione originaria, mentre l'intestazione IP esterna può contenere indirizzi IP diversi (ad esempio, indirizzi di firewall o di altri gateway di sicurezza).

Se si utilizza la modalità tunnel, AH protegge l'intero pacchetto IP interno, compresa la sua intestazione IP. L'intestazione IP esterna e, nel caso di IPv6, le intestazioni di estensione IP esterne sono protette con l'esclusione dei campi modificabili o non prevedibili.

6.4 Encapsulating security payload (ESP)

ESP fornisce servizi volti ad assicurare la riservatezza, inclusa quella del contenuto dei messaggi e una parziale riservatezza del flusso di traffico. Opzionalmente, ESP può anche fornire servizi di autenticazione.

Formato di ESP

La Figura 6.7 mostra il formato di un pacchetto ESP. Il pacchetto contiene i seguenti campi.

- **Indice dei parametri di sicurezza, 32 bit (SPI, security parameters index)**: identifica una SA.
- **Numero di sequenza, 32 bit (sequence number)**: contatore con valori monotonicamente crescenti; fornisce funzionalità anti-replay, come descritto nel caso di AH.
- **Dati di payload, lunghezza variabile (payload data)**: segmento del livello trasporto (se si utilizza la modalità trasporto) oppure pacchetto IP (se si utilizza la modalità tunnel) protetto tramite cifratura.
- **Completamento, da 0 a 255 byte (padding)**: campo di completamento il cui scopo verrà affrontato in seguito.
- **Lunghezza del completamento, 8 bit (pad length)**: indica il numero di byte di completamento del campo precedente.
- **Intestazione successiva, 8 bit (next header)**: identifica la tipologia di dati contenuta nel campo payload, mediante l'identificazione della prima intestazione nel payload in

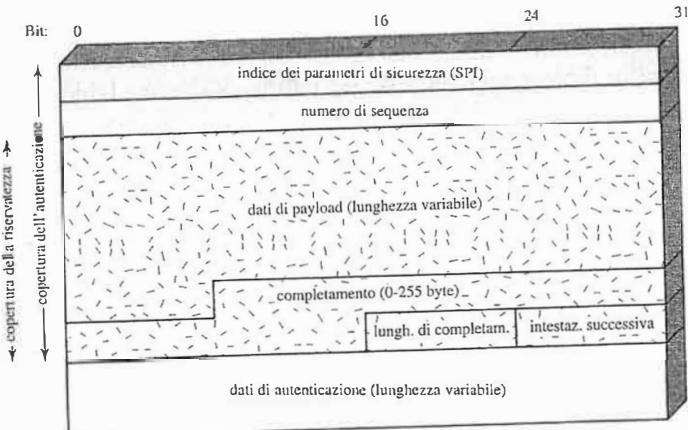


Figura 6.7 Formato di ESP IPSec.

questione (ad esempio, un'intestazione di estensione nel caso di IPv6, o un protocollo di un livello superiore come TCP).

- **Dati di autenticazione, lunghezza variabile (authentication data)**: campo di lunghezza variabile (deve essere un multiplo di parole di 32 bit) contenente l'ICV. Tale valore è calcolato prendendo come base il pacchetto ESP con l'eccezione del campo dati di autenticazione.

Algoritmi di cifratura e autenticazione

I campi payload, lunghezza di completamento e intestazione successiva sono cifrati dal servizio ESP. Nel caso in cui l'algoritmo utilizzato per la cifratura del payload richieda dati per la sincronizzazione critografica – quali, ad esempio, il vettore di inizializzazione (IV, per l'initialization vector) – questi dati possono essere contenuti in modo esplicito all'inizio del campo dati di payload. Se presente, l'IV di solito non è cifrato, sebbene spesso venga considerato come parte del testo cifrato.

La specifica corrente impone che una realizzazione conforme debba comprendere DES in modalità CBC (cfr Capitolo 2). Nel documento relativo a DOI sono stati assegnati alcuni identificatori a una serie di altri algoritmi e tali algoritmi potrebbero quindi essere facilmente utilizzati per la cifratura; di questo insieme di algoritmi fanno parte:

- Triplo DES a tre chiavi
- RC5
- IDEA
- Triplo IDEA a tre chiavi
- CAST
- Blowfish

Analogamente ad AH, ESP fornisce supporto all'utilizzo di un MAC con una lunghezza di default pari a 96 bit. Inoltre, come nel caso di AH, la specifica corrente impone che una realizzazione conforme debba fornire supporto per HMAC-MD5-96 e HMAC-SHA-1-96.

Completamento

Il campo completamento ha molteplici scopi.

- Nel caso in cui un algoritmo di cifratura richieda che il testo in chiaro sia multiplo di un certo numero di byte (ad esempio, un multiplo di un singolo blocco nel caso di cifrari a blocchi), il campo completamento è utilizzato per espandere il testo in chiaro (che consiste nei campi dati di payload, completamento, lunghezza di completamento e intestazione successiva) fino a raggiungere la lunghezza richiesta.
- Il formato ESP richiede che i campi lunghezza di completamento e intestazione successiva siano allineati a destra all'interno di una parola di 32 bit. Equivalentemente, il testo cifrato deve essere un multiplo intero di 32 bit. Il campo completamento è utilizzato per assicurare tale allineamento.
- Ulteriori completamenti possono essere aggiunti per fornire una parziale riservatezza del flusso di traffico, così da non rivelare la lunghezza effettiva del payload.

Modalità trasporto e tunnel

La Figura 6.8 illustra due modalità di utilizzo del servizio ESP di IPSec. Nella parte superiore della figura, la cifratura e, optionalmente, l'autenticazione sono fornite direttamente tra due host. La Figura 6.8b illustra come la modalità tunnel possa essere utilizzata per instaurare una **rete privata virtuale**. In questo esempio, un'organizzazione possiede quattro reti private connesse tramite Internet. Gli host posti sulle reti interne utilizzano Internet per il trasporto dati ma non interagiscono con gli altri host connessi ad Internet. Se si utilizza una configurazione in cui i tunnel terminano ai gateway di sicurezza posti su ciascuna rete interna, non è necessario che gli host implementino le funzionalità di sicurezza. La prima modalità richiede l'utilizzo di una SA in modalità trasporto, mentre la seconda tecnica utilizza una SA in modalità tunnel.

In questo paragrafo si analizzerà il campo di azione di ESP per entrambe le modalità. Le considerazioni effettuate saranno in alcuni casi diverse per IPv4 e IPv6. Come nel caso di AH, si utilizzeranno come punto di partenza i formati di pacchetto illustrati nella Figura 6.6a.

ESP in modalità trasporto

ESP in modalità trasporto è utilizzato per cifrare e, optionalmente, autenticare i dati trasmessi tramite IP (ad esempio, un segmento TCP), come illustrato nella Figura 6.9a. Quando si utilizza ESP in modalità trasporto in IPv4, l'intestazione ESP è collocata nel pacchetto IP immediatamente prima dell'intestazione del livello trasporto (ad esempio: TCP, UDP, ICMP), mentre i campi completamento, lunghezza di completamento e intestazione successiva, sono collocati dopo il pacchetto IP (tali campi vengono indicati con il nome di **trailer ESP**). Se viene scelta l'autenticazione, il campo dati di autenticazione di ESP è aggiunto dopo i campi precedentemente menzionati. Sia l'intero segmento a li-

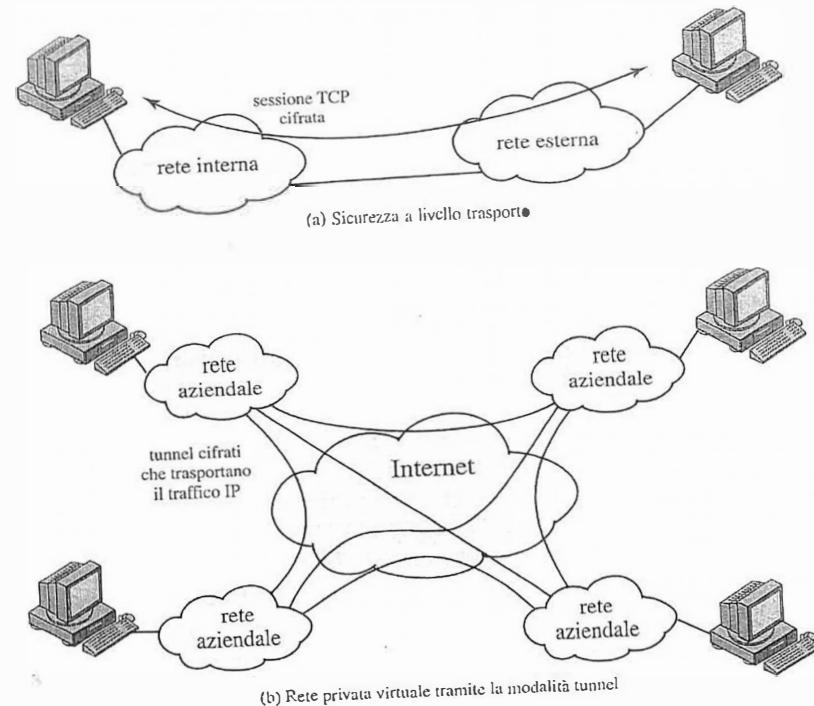


Figura 6.8 Confronto tra la cifratura in modalità trasporto e in modalità tunnel.

vello trasporto sia i campi sopra descritti sono cifrati. L'autenticazione copre tutto il testo cifrato ed anche l'intestazione ESP.

Per quanto riguarda IPv6, ESP è visto come un payload end-to-end. Ciò implica che i router intermedi non effettuano alcuna elaborazione o esame. L'intestazione ESP è quindi collocata dopo l'intestazione base IPv6 e dopo le intestazioni di estensione hop-by-hop, di destradamento e di frammentazione. L'intestazione di estensione relativa alle opzioni di destinazione può essere collocata sia prima sia dopo l'intestazione ESP, in dipendenza dal significato che si vuole attribuire. Nel caso di IPv6, la cifratura copre, oltre che l'intero segmento a livello di trasporto e il trailer ESP, anche l'intestazione di estensione relativa alle opzioni di destinazione, nel caso in cui tale intestazione compaia dopo l'intestazione ESP. Anche in questo caso, l'autenticazione copre sia il testo cifrato sia l'intestazione ESP.

Il funzionamento della modalità trasporto può essere riassunto come segue.

1. Alla sorgente si effettua la cifratura del blocco di dati composto dal trailer ESP e dall'intero segmento a livello trasporto. Il testo in chiaro corrispondente a questo blocco è sostituito dal corrispondente testo cifrato per formare il pacchetto IP da trasmettere. Viene aggiunta l'autenticazione, se la corrispondente opzione è stata selezionata.

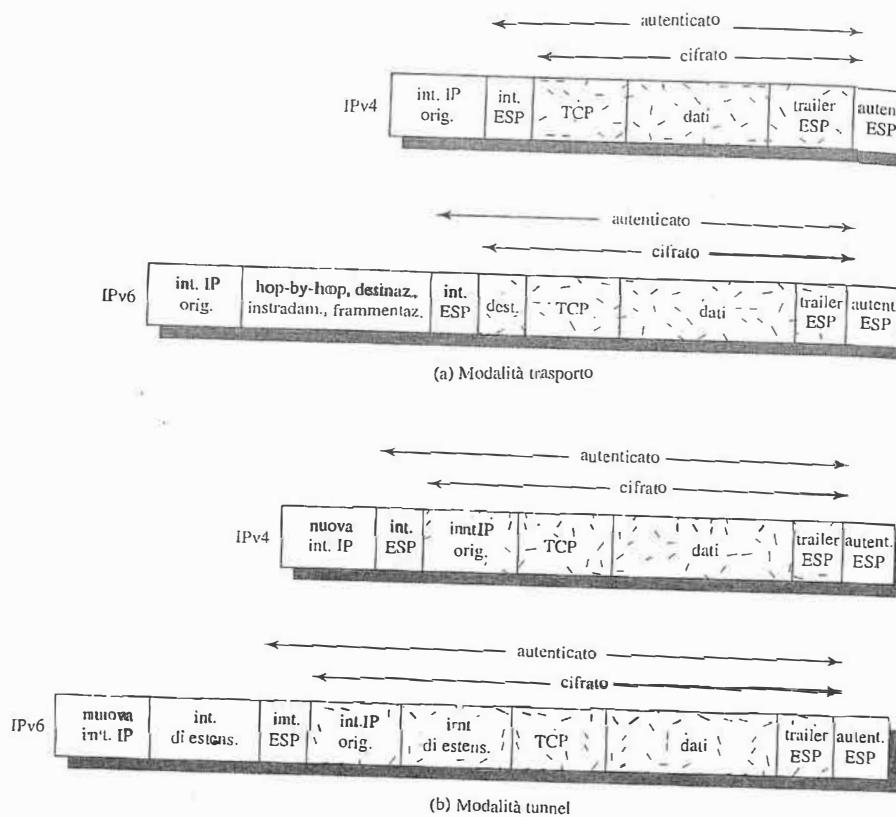


Figura 6.9 Ambito della cripitura e dell'autenticazione ESP.

- Il pacchetto è successivamente instradato a destinazione. Ogni router intermedio ha bisogno di esaminare ed elaborare sia l'intestazione IP sia ciascuna intestazione di estensione IP trasmessa in chiaro, ma non ha bisogno di esaminare il testo cifrato.
- Il nodo destinazione esamina ed elabora sia l'intestazione IP sia tutte le intestazioni di estensione IP in chiaro. Successivamente, sulla base di SPI contenuto nell'intestazione ESP, il nodo destinazione decifra la rimanente parte del pacchetto al fine di ricostruire il segmento di livello trasporto in chiaro.

La modalità trasporto fornisce riservatezza per ogni applicazione che la utilizza, evitando, quindi, la necessità di realizzare in ogni singola applicazione meccanismi che assicurino riservatezza. Tale modo di operare è anche ragionevolmente efficiente, in quanto non altera di molto la lunghezza complessiva del pacchetto IP. Uno degli svantaggi della modalità trasporto è che consente di effettuare analisi del traffico sui pacchetti trasmessi.

ESP in modalità tunnel

ESP in modalità tunnel è utilizzato per cifrare un intero pacchetto IP (Figura 6.9b). In tale modalità, l'intestazione ESP precede il pacchetto. Il pacchetto e il trailer ESP vengono poi cifrati. Questo metodo può essere usato per rendere difficile l'analisi del traffico.

Dal momento che l'intestazione IP contiene l'indirizzo di destinazione e possibilmente anche le direttive per l'instradamento dalla sorgente e le informazioni riguardanti le opzioni hop-by-hop, non è possibile trasmettere semplicemente il pacchetto IP cifrato preceduto dall'intestazione ESP, in quanto i router intermedi non sarebbero in grado di gestire tale pacchetto. È dunque necessario incapsulare l'intero blocco (cioè l'intestazione ESP, il testo cifrato e il campo dati di autenticazione, se presente) con una nuova intestazione IP che conterrà informazioni sufficienti ad effettuare l'instradamento ma non l'analisi del traffico.

Mentre la modalità trasporto è adatta alla protezione di connessioni tra host che supportano ESP, la modalità tunnel è consigliabile nel caso di configurazioni che contengano un firewall o altre tipologie di gateway di sicurezza che proteggono una rete fidata da reti esterne. In quest'ultimo caso, la cripitura ha luogo solo tra un host esterno e il gateway di sicurezza, oppure tra due gateway di sicurezza. Questo schema evita agli host sulla rete interna il carico dovuto alla cripitura e semplifica il compito di distribuzione delle chiavi riducendo il numero di chiavi necessarie. Tale schema rende, inoltre, difficile l'analisi del traffico basata sulla destinazione finale.

Si consideri il caso di un host esterno che desidera comunicare con un host su una rete interna protetta tramite firewall. Si assuma, inoltre, che ESP sia implementato sull'host esterno e sul firewall. Per effettuare il trasferimento di un segmento di livello trasporto dall'host esterno all'host interno, sono necessari i seguenti passi.

- La sorgente prepara un pacchetto IP interno avente come indirizzo di destinazione quello dell'host interno cui si vuole spedire il messaggio. Prima del pacchetto, viene aggiunta un'intestazione ESP; il pacchetto e il trailer ESP vengono quindi cifrati e, optionalmente, può essere aggiunto il campo dati di autenticazione. Il blocco risultante è incapsulato con una nuova intestazione IP (intestazione base più estensioni facoltative come quelle relative alle opzioni di instradamento e all'hop-by-hop nel caso di IPv6) avente come indirizzo di destinazione quello del firewall; tale blocco costituisce il pacchetto IP esterno.
- Il pacchetto esterno è instradato al firewall destinazione. Ogni router intermedio ha bisogno di esaminare ed elaborare sia l'intestazione IP esterna sia ogni intestazione di estensione IP esterna, ma non ha bisogno di esaminare il testo cifrato.
- Il firewall destinazione esamina ed elabora l'intestazione IP esterna e tutte le intestazioni di estensione IP esterne. Successivamente il nodo destinazione decifra, sulla base di SPI contenuto nell'intestazione ESP, la parte rimanente del pacchetto per ricostruire il pacchetto IP interno in chiaro. Tale pacchetto è poi trasmesso sulla rete interna.
- Il pacchetto interno è instradato sulla rete interna verso l'host destinazione, attraverso zero o più router.

6.5 Combinazione di più SA

Una singola SA può realizzare il protocollo AH o il protocollo ESP, ma non entrambi. Esistono però alcuni casi, relativi a flussi di traffico particolari, in cui può sorgere la necessità di disporre sia dei servizi offerti da AH sia di quelli forniti da ESP. Inoltre, un particolare flusso di traffico può richiedere servizi IPSec tra gli host e, allo stesso tempo, servizi distinti tra i gateway di sicurezza come, ad esempio, i firewall. In questi casi, si possono impiegare più SA per lo stesso flusso di traffico al fine di ottenere i servizi IPSec desiderati. Con il termine **bundle di SA** si intende una sequenza di SA tramite le quali il traffico deve essere elaborato al fine di ottenere l'insieme di servizi IPSec richiesti. Le SA che compongono un bundle possono terminare tutte allo stesso punto finale, oppure terminare in punti differenti.

Per formare un bundle, le SA possono essere combinate secondo due modalità.

- **Transport adjacency** (adiacenza di trasporto): consiste nell'applicare allo stesso pacchetto IP più di un protocollo di sicurezza, senza utilizzare la modalità tunnel. Questo approccio per combinare AH ed ESP consente un solo livello di combinazione; ulteriori annidamenti non portano alcun beneficio, in quanto l'elaborazione è già effettuata da una sola istanza IPSec, alla destinazione finale.
- **Iterate tunneling** (modalità tunnel iterata): in base a tale modalità vengono applicati più livelli di protocolli di sicurezza realizzati tramite la modalità tunnel IP. Questo approccio consente più livelli di annidamento, dal momento che ogni tunnel può aver origine o terminare a un diverso sito IPSec posto lungo il percorso. I due approcci possono essere combinati (ad esempio, utilizzando una SA in modalità trasporto tra gli host, che effettua parte del percorso tramite una SA in modalità tunnel stabilita tra i gateway di sicurezza).

Quando si considerano bundle di SA, sorgono interessanti questioni riguardo l'ordine in cui la cifratura e l'autenticazione possono essere applicate tra una data coppia di punti, e il metodo per effettuarle. In seguito, si esamineranno queste problematiche e si considereranno combinazioni di SA che coinvolgono almeno un tunnel.

Autenticazione e riservatezza

Cifratura e autenticazione possono essere combinate al fine di trasmettere un pacchetto IP che assicuri sia la riservatezza sia l'autenticazione tra host. In seguito verranno considerati diversi approcci.

ESP con opzione di autenticazione

In questa tipologia di approccio (Figura 6.9), l'utente applica inizialmente ESP ai dati da proteggere e successivamente accoda il campo dati di autenticazione. ESP con opzione di autenticazione presenta due sottocasi.

- **ESP in modalità trasporto:** l'autenticazione e la cifratura si applicano al payload IP inviato all'host, mentre l'intestazione IP non viene protetta.
- **ESP in modalità tunnel:** l'autenticazione si applica all'intero pacchetto IP inviato all'indirizzo IP esterno (ad esempio, un firewall), e l'autenticazione è effettuata presso

tal destinazione. L'intero pacchetto IP interno è protetto dal meccanismo di riservatezza, per la consegna alla destinazione IP interna.

In entrambi i casi, l'autenticazione si applica al testo cifrato e non al testo in chiaro.

Transport adjacency

Un altro modo di effettuare l'autenticazione dopo la cifratura è utilizzare un bundle costituito da due SA in modalità trasporto, dove la SA interna è una SA ESP, mentre quella esterna è una SA AH. In questo caso ESP viene utilizzato senza opzione di autenticazione. Dato che la SA interna è in modalità trasporto, la cifratura è applicata al payload IP. Il pacchetto che si ottiene consiste in un'intestazione IP (ed, eventualmente, in intestazioni di estensione IPv6) seguita da un ESP. AH è quindi applicato in modalità trasporto, in modo che l'autenticazione copra sia l'ESP sia l'intestazione IP originaria (e le estensioni) con l'eccezione dei campi modificabili. Il vantaggio di tale approccio, rispetto al semplice utilizzo di un'unica SA ESP con l'opzione di autenticazione ESP, è che – in questo caso – l'autenticazione copre più campi, inclusi gli indirizzi IP della sorgente e della destinazione. Lo svantaggio è il costo addizionale in quanto si hanno due SA invece di una.

Bundle trasporto-tunnel

L'utilizzo dell'autenticazione prima di effettuare la cifratura potrebbe essere vantaggioso per svariati motivi. In primo luogo, essendo i dati di autenticazione protetti dalla cifratura, è impossibile che qualcuno intercetti il messaggio e modifichi tali dati senza che questo venga rilevato. In secondo luogo, in certi casi può essere utile memorizzare alla destinazione le informazioni di autenticazione congiuntamente al messaggio, per riferimenti futuri. È più conveniente fare ciò se le informazioni di autenticazione si riferiscono al messaggio non cifrato; in caso contrario sarebbe necessario cifrare nuovamente il messaggio per verificare le informazioni di autenticazione.

Un approccio per applicare l'autenticazione prima della cifratura tra due host consiste nell'utilizzare un bundle composto da una SA AH interna in modalità trasporto e da una SA ESP esterna in modalità tunnel. In questo caso, l'autenticazione è applicata sia al payload IP sia all'intestazione IP (e alle estensioni) ad esclusione dei campi modificabili. Il pacchetto IP risultante è quindi elaborato in modalità tunnel dall'ESP; il risultato è che l'intero pacchetto interno autenticato viene cifrato, e viene aggiunta una nuova intestazione IP esterna (e le estensioni).

Combinazioni base di SA

Il documento relativo all'architettura di IPSec elenca quattro esempi di combinazioni di SA che devono essere fornite da tutti gli host conformi alle specifiche IPSec (ad esempio, una workstation o un server) o dai gateway di sicurezza (ad esempio, firewall, router). Tali esempi sono illustrati nella Figura 6.10. La parte inferiore della porzione di figura relativa a ogni tipologia di esempio illustra la connettività fisica degli elementi; la parte superiore rappresenta, invece, la connettività logica ottenuta tramite una o più SA annidate. Ciascuna SA può essere AH oppure ESP. Nel caso di SA tra host, la modalità può essere sia tunnel sia trasporto; in tutti gli altri casi, deve essere utilizzata la modalità tunnel.

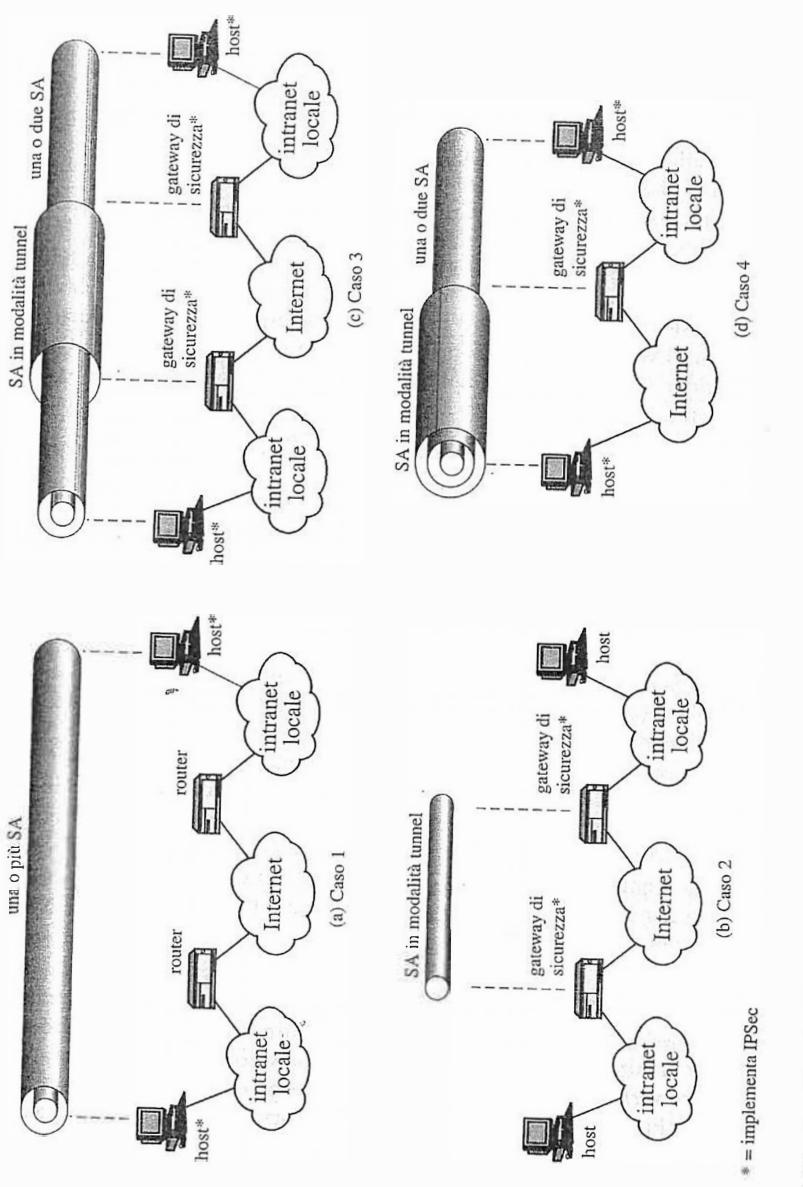


Figura 6.10 Combinazioni base di SA

Nel **Caso 1**, la sicurezza è fornita tra i sistemi terminali che implementano IPsec. Affinché due sistemi terminali possano comunicare tramite una SA, devono condividere le chiavi segrete appropriate. Vediamo alcune delle possibili combinazioni.

- AH in modalità trasporto.
- ESP in modalità trasporto.
- ESP seguito da AH in modalità a trasporto (una SA ESP all'interno di una SA AH).
- Uno qualsiasi tra i casi a, b, o c all'interno di una SA AH o ESP in modalità tunnel.

Si è già visto come le varie combinazioni sopra illustrate possano essere utilizzate per fornire autenticazione, cifratura ed autenticazione prima o dopo la cifratura.

Nel **Caso 2**, la sicurezza è fornita soltanto tra gateway (router, firewall, etc.), mentre gli host non implementano IPsec. Questo caso illustra il supporto fornito per semplici reti virtuali private. Il documento relativo all'architettura di sicurezza specifica che, in questo caso, è sufficiente una singola SA in modalità tunnel. Il tunnel potrebbe fornire il supporto per AH, ESP oppure ESP con l'opzione di autenticazione. Non sono necessari tunnel annidati, dal momento che i servizi IPsec si applicano all'intero pacchetto interno.

Il **Caso 3** si basa sul Caso 2 aggiungendo a questo la sicurezza end-to-end. Qui sono permesse le stesse combinazioni viste nei Casi 1 e 2. Il tunnel che collega i gateway fornisce autenticazione o riservatezza, oppure entrambe, per tutto il traffico tra i due sistemi terminali. Nel caso in cui il tunnel che connette i gateway sia un tunnel ESP, viene fornita anche una parziale riservatezza del traffico. I singoli host possono realizzare, tramite SA end-to-end, qualsiasi ulteriore servizio IPsec richiesto da particolari applicazioni o utenti.

Il **Caso 4** fornisce supporto per la gestione di un host remoto che utilizza Internet per raggiungere il firewall di un'organizzazione e ottenere quindi l'accesso a una workstation o a server protetti dal firewall. È richiesta solo la modalità tunnel tra l'host remoto e il firewall. Come nel Caso 1, possono essere utilizzate una oppure due SA tra l'host remoto e quello locale.

6.6 Gestione delle chiavi

La parte di IPsec relativa alla gestione delle chiavi affronta il problema sia della scelta sia della distribuzione delle chiavi segrete. Una necessità comune è quella di poter disporre di quattro chiavi (cioè una coppia di chiavi per la trasmissione e la ricezione tanto per AH che per ESP) da utilizzare per la comunicazione tra due applicazioni. Il documento che descrive l'architettura IPsec prevede due tipi di gestione delle chiavi,

- Gestione manuale:** l'amministratore di sistema configura manualmente ciascun sistema con le proprie chiavi e con le chiavi dei sistemi che devono comunicare con esso. Tale approccio è realizzabile nel caso di ambienti piccoli e relativamente statici.
- Gestione automatizzata:** il sistema automatico consente di creare, su richiesta, le chiavi per le SA. Tale approccio facilita l'utilizzo delle chiavi in sistemi distribuiti di grosse dimensioni con una configurazione che si evolve nel tempo.

Il protocollo di default per la gestione automatizzata delle chiavi in IPsec è chiamato ISAKMP/Oakley e si compone dei seguenti elementi:

- **Protocollo Oakley** per la scelta delle chiavi. Oakley è un protocollo per lo scambio di chiavi basato sull'algoritmo Diffie-Hellman ma rispetto a quest'ultimo fornisce maggiori garanzie di sicurezza. Oakley è un protocollo generale in quanto non impone alcuno specifico formato per le chiavi.
- **Protocollo Internet** per le SA e la gestione delle chiavi (ISAKMP, *Internet security association and key management protocol*). ISAKMP fornisce sia un'architettura di riferimento per la gestione delle chiavi in Internet sia il supporto per uno specifico protocollo – formati inclusi – per la negoziazione degli attributi di sicurezza.

ISAKMP non impone alcuno specifico algoritmo per lo scambio di chiavi, ma è composto da un insieme di tipologie di messaggi che rendono possibile l'utilizzo di una varietà di algoritmi per lo scambio di chiavi. Oakley rappresenta la scelta obbligatoria come algoritmo per lo scambio di chiavi per la versione iniziale di ISAKMP.

Protocollo Oakley per la scelta delle chiavi

Oakley costituisce un raffinamento dell'algoritmo Diffie-Hellman per lo scambio di chiavi. Si ricordi come l'algoritmo Diffie-Hellman comporti le seguenti interazioni tra due utenti A e B. Viene effettuato un accordo preliminare su due parametri globali: q , un numero primo di dimensioni elevate, ed α , una radice primitiva di q . A sceglie, come sua chiave privata, un intero casuale X_A e invia a B la propria chiave pubblica $Y_A = \alpha^{X_A} \text{ mod } q$. In maniera analoga, B sceglie un intero casuale X_B , come sua chiave privata, e invia ad A la propria chiave pubblica $Y_B = \alpha^{X_B} \text{ mod } q$. Entrambe le parti possono a questo punto calcolare la chiave segreta di sessione:

$$K = (Y_B)^{X_A} \text{ mod } q = (Y_A)^{X_B} \text{ mod } q = \alpha^{X_A X_B} \text{ mod } q$$

L'algoritmo Diffie-Hellman presenta due interessanti caratteristiche.

- Le chiavi segrete sono create solo quando effettivamente necessarie. Non occorre memorizzare chiavi segrete per lunghi periodi, cosa che le renderebbe maggiormente vulnerabili.
- Lo scambio non richiede alcuna infrastruttura pre-esistente ad eccezione dell'accordo sui parametri globali.

Nonostante tali vantaggi, esistono, comunque, nell'algoritmo Diffie-Hellman numerosi punti deboli, rilevati in [HUIT98].

- L'algoritmo non fornisce alcuna informazione circa l'identità delle parti coinvolte.
- L'algoritmo è vulnerabile ad attacchi di tipo *man-in-the-middle* (uomo nel mezzo), nei quali una terza parte E impersona B durante la comunicazione con A e, nel contempo, impersona A durante la comunicazione con B. Sia A che B negozierebbero, quindi, una chiave con E che può, di conseguenza, ascoltare e trasmettere traffico sulla rete. Il tipo di attacco *man-in-the-middle* si attua secondo le seguenti modalità.
 1. B invia la propria chiave pubblica Y_B in un messaggio indirizzato ad A (Figura 3.11).
 2. L'avversario (E) intercetta il messaggio, salva la chiave pubblica di B e invia un messaggio ad A che contiene come ID utente quello di B e come chiave pubblica Y_E . Il messaggio è trasmesso in modo tale che appaia come se fosse stato

mandato dal sistema host di B. A riceve il messaggio di E e memorizza la chiave pubblica di E associandovi l'ID di B. In modo analogo, E invia a B un messaggio che contiene la propria chiave pubblica in modo che sembri provenire da A.

3. B calcola una chiave segreta K_1 basata sulla chiave privata di B e su Y_E . A calcola una chiave segreta K_2 , sulla base della chiave privata di A e di Y_E . E calcola K_1 , utilizzando la chiave segreta di E (X_E) e Y_B , e calcola K_2 utilizzando X_E e Y_A .
4. Da questo momento, E è in grado di controllare i messaggi da A a B e da B ad A, modificando appropriatamente durante la trasmissione la loro cifratura in modo che né A né B si rendano conto del fatto che stanno condividendo le loro comunicazioni con E.
- È computazionalmente molto oneroso. Per tale motivo, è vulnerabile a un attacco cosiddetto di *clogging* (saturazione), in cui un avversario richiede un elevato numero di chiavi. In questo modo, la vittima dell'attacco impegnava notevoli risorse computazionali effettuando inutili elevamenti a potenza e operazioni di modulo.

Oakley è progettato per mantenere i vantaggi dell'algoritmo Diffie-Hellman superandone nel contempo le debolezze.

Caratteristiche di Oakley

L'algoritmo Oakley è contraddistinto da cinque importanti caratteristiche.

1. Utilizza il meccanismo dei cookie per contrastare attacchi di clogging.
2. Consente alle due parti in causa di negoziare un gruppo che sostanzialmente specifica i parametri globali dello scambio di chiavi mediante Diffie-Hellman.
3. Utilizza dei nonce per rendere inefficaci gli attacchi di replay.
4. Consente lo scambio Diffie-Hellman di chiavi pubbliche.
5. Autentica lo scambio Diffie-Hellman per contrastare attacchi di tipo *man-in-the-middle*.

Lo schema di Diffie-Hellman è già stato illustrato nei capitoli precedenti. In questa sede, si analizzeranno pertanto, uno alla volta, i rimanenti elementi di Oakley. Si consideri, in primo luogo, il problema degli attacchi di clogging. In questo tipo di attacchi, l'indirizzo sorgente di un utente legittimo viene contraffatto e alla vittima è inviata una chiave pubblica Diffie-Hellman. La vittima effettua, quindi, un elevamento a potenza seguito da un'operazione di modulo per calcolare la chiave segreta. Messaggi ripetuti di questo tipo possono "saturare" il sistema della vittima con lavoro inutile. Lo **scambio di cookie** richiede che ciascuna parte invii un numero pseudo-casuale – detto cookie – nel messaggio iniziale, e che l'altra parte ne comunichi l'avvenuta ricezione. La segnalazione di avvenuta ricezione deve essere ripetuta nel primo messaggio dello scambio di chiavi Diffie-Hellman. Nel caso in cui l'indirizzo sorgente sia stato contraffatto, l'avversario non riceverà risposta. Un avversario può solo costringere un utente a comunicare l'avvenuta ricezione ma non ad eseguire l'algoritmo Diffie-Hellman.

ISAKMP impone che la generazione dei cookie soddisfi tre requisiti base.

1. Il cookie deve dipendere dalle specifiche parti coinvolte. Tale requisito impedisce a un avversario di ottenere un cookie usando un indirizzo IP e una porta UDP reali e di utilizzare successivamente il cookie per sommersa la vittima di richieste originate da indirizzi IP o porte scelte casualmente.
2. Il meccanismo di generazione dei cookie deve assicurare che l'entità che riceve il cookie lo accetti solo dall'entità che lo ha per lei generato. Ciò comporta che quest'ultima utilizza informazioni segrete locali nel processo di generazione e di successiva verifica di un cookie. Non deve essere possibile dedurre queste informazioni segrete dai cookie. Il punto fondamentale di questo requisito è che l'entità generatrice non ha bisogno di salvare copie dei suoi cookie, che potrebbero essere in questo modo più facilmente vulnerabili, ma può allo stesso tempo verificare una notifica di avvenuta ricezione di un cookie, in caso di necessità.
3. I metodi per la generazione e la verifica dei cookie devono essere veloci per contrastare attacchi finalizzati a sabotare le risorse di elaborazione.

Il metodo raccomandato per la creazione dei cookie consiste nel computare un'efficiente funzione hash (per esempio MD5) sugli indirizzi IP della sorgente e della destinazione, sulle porte UDP della sorgente e della destinazione e su un valore segreto generato localmente.

Oakley fornisce la possibilità di utilizzare differenti gruppi per lo scambio di chiavi Diffie-Hellman. Ciascun gruppo specifica i due parametri globali e l'algoritmo da utilizzare. L'attuale specifica comprende i seguenti gruppi:

- elevamento a potenza seguito da un'operazione di modulo con modulo di 768 bit
 $q = 2^{768} - 2^{704} - 1 + 2^{64} \times ([2^{638} \times \pi] + 149686)$
 $\alpha = 2$
- elevamento a potenza seguito da un'operazione di modulo con modulo di 1024 bit
 $q = 2^{1024} - 2^{960} - 1 + 2^{64} \times ([2^{894} \times \pi] + 129093)$
 $\alpha = 2$
- elevamento a potenza seguito da un'operazione di modulo con modulo di 1536 bit
 - parametri da determinare
- gruppo delle curve ellittiche su 2^{155}
 - generatore (esadecimale): X = 7B, Y = 1C8
 - parametri delle curve ellittiche (esadecimale): A=0, Y=7338F
- gruppo delle curve ellittiche su 2^{185}
 - generatore (esadecimale): X = 18, Y = D
 - parametri delle curve ellittiche (esadecimale): A = 0, Y = 1EE9

I primi tre gruppi si riferiscono all'algoritmo Diffie-Hellman classico, che utilizza le operazioni di elevamento a potenza e modulo. Gli ultimi due gruppi utilizzano le curve ellittiche.

Per prevenire attacchi di replay, Oakley utilizza i **nonce**, numeri pseudo-casuali generati localmente. Questi sono presenti nelle risposte e sono cifrati durante alcune parti dello scambio per rendere sicuro il loro utilizzo.

Con Oakley possono essere utilizzati tre differenti metodi di autenticazione.

- **Firme digitali:** lo scambio è autenticato firmando un valore hash ottenibile reciprocamente; ciascuna parte cifra il valore hash con la sua chiave privata. Il valore hash è generato considerando parametri significativi quali gli ID utente e i nonce.
- **Cifratura a chiave pubblica:** lo scambio è autenticato mediante la cifratura, con la chiave privata del mittente, di parametri quali gli ID e i nonce.
- **Cifratura a chiave simmetrica:** una chiave, derivata con un qualche meccanismo fuori banda, può essere utilizzata per autenticare gli scambi tramite la cifratura simmetrica dei parametri di scambio.

Esempio di scambi mediante Oakley

La specifica di Oakley contiene un certo numero di esempi di scambi consentiti in base a tale protocollo. Per dare un'idea di Oakley, verrà presentato un esempio che nella specifica è chiamato scambio di chiavi aggressive, perché vengono utilizzati solo tre messaggi.

La Figura 6.11 illustra il relativo protocollo. Nella prima fase, colui che inizia il protocollo (I) invia un cookie, il gruppo da utilizzare e la sua chiave pubblica Diffie-Hellman per lo scambio in questione. I indica anche la cifratura a chiave pubblica proposta e gli algoritmi hash e di autenticazione da utilizzare in questo scambio. Nel messaggio sono inoltre contenuti gli identificatori di I e del ricevente (R) e il nonce utilizzato da I per lo scambio in questione. Infine, I appone una firma utilizzando la sua chiave privata per firmare i due identificatori, il nonce, il gruppo, la chiave pubblica Diffie-Hellman e gli algoritmi proposti.

Quando R riceve il messaggio, verifica la firma utilizzando la chiave pubblica di firma di I. R segnala l'avvenuta ricezione del messaggio rispedendo a I il cookie, l'identificatore, il nonce

I → R: CKY _I , OK_KEYX, GRP, g ^x , EHAO, NIDP, ID _I , ID _R , N _I , S _{KI} [ID _I ID _R N _I GRP g ^x EHAO]
R → I: CKY _R , CKY _I , OK_KEYX, GRP, g ^y , EHAS, NIDP, ID _R , ID _I , N _R , N _I , S _{KR} [ID _R ID _I N _R N _I GRP g ^y EHAS]
I → R: CKY _I , CKY _R , OK_KEYX, GRP, g ^x , EHAS, NIDP, ID _I , ID _R , N _I , N _R , S _{KI} [ID _I ID _R N _I N _R GRP g ^x EHAS]

Legenda:

- I = iniziatore
- R = ricevente
- CKY_I, CKY_R = cookie dell'iniziatore e del ricevente
- OK_KEYX = tipologia di messaggio per scambio di chiavi
- GRP = nome del gruppo Diffie-Hellman per lo scambio in questione
- g^x, g^y = chiave pubblica dell'iniziatore e del ricevente; g^{xy} = chiave di sessione derivata da g^x e g^y per questo scambio
- EHAO, EHAS = cifratura, hash, funzioni di autenticazione, proposte e scelte
- NIDP = indica che non viene usata la cifratura per la rimanente parte di messaggio
- ID_I, ID_R = identificatore dell'iniziatore e del ricevente
- N_I, N_R = nonce casuale fornito dall'iniziatore e dal ricevente per lo scambio in questione
- S_{KI}[X], S_{KR}[X] = firma di X con la chiave privata (chiave di firma) dell'iniziatore e del ricevente

Figura 6.11 Esempio di scambio delle chiavi aggressive mediante Oakley.

e il gruppo contenuti nel messaggio ricevuto da I. Nel messaggio, R include anche un cookie, la sua chiave pubblica Diffie-Hellman, gli algoritmi scelti (che devono essere tra gli algoritmi proposti da I), il proprio identificatore e il proprio nonce per lo scambio in questione. Infine, R appone una firma utilizzando la propria chiave privata per firmare i due identificatori, i due nonce, il gruppo, le due chiavi pubbliche Diffie-Hellman e gli algoritmi scelti.

Quando I riceve il secondo messaggio, verifica la firma utilizzando la chiave pubblica di R. Il valore del nonce contenuto nel messaggio assicura che non si tratta di un attacco di replay. Per completare lo scambio, I deve spedire un messaggio a R per attestare di aver ricevuto la chiave pubblica di R.

ISAKMP

ISAKMP stabilisce le procedure e i formati dei pacchetti per instaurare, negoziare, modificare e cancellare SA. Come parte della creazione di una SA, ISAKMP definisce i payload per lo scambio di dati relativi alla generazione di chiavi e all'autenticazione. Il formato dei payload fornisce un'architettura di riferimento consistente, indipendente dallo specifico protocollo usato per lo scambio delle chiavi, dall'algoritmo di cifratura e dal meccanismo di autenticazione.

Formato dell'intestazione ISAKMP

Un messaggio ISAKMP è composto da un'intestazione ISAKMP seguita da uno o più payload. Tutto ciò è trasmesso mediante un protocollo di livello trasporto. La specifica stabilisce che le implementazioni debbano fornire il supporto per l'utilizzo di UDP come protocollo di livello trasporto.

La Figura 6.12a mostra il formato dell'intestazione per un messaggio ISAKMP. Tale intestazione è composta dai seguenti campi.

- **Cookie dell'iniziatore, 64 bit (*initiator cookie*)**: cookie relativo all'entità che ha dato inizio alla generazione, notifica o cancellazione della SA.
- **Cookie del ricevente, 64 bit (*responder cookie*)**: cookie dell'entità che risponde; tale campo è nullo nel primo messaggio mandato dall'iniziatore.
- **Payload successivo, 8 bit (*next payload*)**: indica il tipo del primo payload contenuto nel messaggio. I payload saranno illustrati successivamente in questo paragrafo.
- **Versione principale, 4 bit (*major version*)**: indica la versione principale di ISAKMP in uso.
- **Versione minore, 4 bit (*minor version*)**: indicata la versione minore attualmente in uso.
- **Tipologia di scambio, 8 bit (*exchange type*)**: indica il tipo di scambio (trattato successivamente).
- **Flag, 8 bit (*flag*)**: indica le specifiche opzioni fissate per lo scambio ISAKMP in questione. Finora sono stati definiti due bit: il bit di cifratura – che viene inizializzato se tutti i payload che seguono l'intestazione sono cifrati mediante l'algoritmo di cifratura relativo alla SA in questione – e il bit di commit, utilizzato per assicurare che ciò che viene cifrato non sia ricevuto prima che la SA sia instaurata.

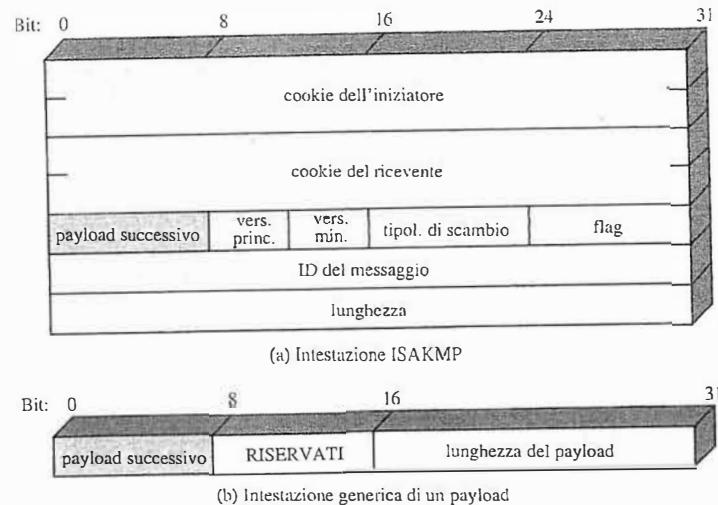


Figura 6.12 Formati ISAKMP.

- **ID del messaggio, 32 bit (*message ID*)**: identifica in modo univoco il messaggio in questione.
- **Lunghezza, 32 bit (*length*)**: lunghezza dell'intero messaggio (intestazione più tutti i payload) espressa in byte.

Tipologie di payload ISAKMP

Tutti i payload ISAKMP iniziano con la stessa intestazione generica, illustrata nella Figura 6.12b. Il campo payload successivo assume valore 0 se quello in questione è l'ultimo payload nel messaggio; altrimenti il valore di tale campo è costituito dal tipo del payload successivo. Il campo lunghezza del payload indica la lunghezza in byte del payload considerato, inclusa l'intestazione generica.

La Tabella 6.3 riassume le tipologie di payload definite per ISAKMP ed elenca i campi, o parametri, che sono parte di ciascun payload. Il payload SA è utilizzato per iniziare l'instaurazione di una SA. In questo tipo di payload, il parametro dominio di interpretazione identifica il DOI (domain of interpretation) in cui ha luogo la negoziazione. Il DOI di IPSec è uno dei possibili esempi, ma ISAKMP può anche essere utilizzato in altri contesti. Il parametro contesto (*situation*) definisce la politica di sicurezza per la negoziazione in questione; in sostanza, vengono specificati i livelli di sicurezza richiesti per la cifratura e la riservatezza (per esempio, livello di sensibilità e categorie di sicurezza).

Il payload proposta (*proposal*) contiene informazioni che vengono utilizzate durante la negoziazione di una SA. Il payload indica il protocollo per questa SA (ESP o AH), di cui si stanno negoziando servizi e meccanismi. Il payload include, inoltre, SPI dell'entità mittente e il numero di trasformazioni. Ogni trasformazione è contenuta in un payload trasformazione.

Tipologia	Parametri	Descrizione
Associazione di sicurezza (SA)	Dominio di interpretazione (DOI), contesto	Utilizzato per la negoziazione di attributi di sicurezza e per specificare il DOI e il contesto in cui ha luogo la negoziazione.
Proposta (P)	# ¹ proposta, ID del protocollo, dimensione SPI, # ¹ di trasformazioni, SPI	Utilizzato durante la negoziazione di una SA; specifica il protocollo da utilizzare ed il numero di trasformazioni.
Trasformazione (T)	# ¹ trasformazione; ID della trasformazione, attributi della SA	Utilizzato durante la negoziazione di una SA; specifica la trasformazione e i relativi attributi della SA.
Scambio di chiavi (KE)	Dati per lo scambio di chiavi	Fornisce supporto ad una varietà di tecniche di scambio di chiavi.
Identificazione (ID)	Tipologia di ID, dati di identificazione	Utilizzato per lo scambio di informazioni di identificazione.
Certificato (CERT)	Codifica del certificato, dati del certificato	Utilizzato per trasferire certificati e altre informazioni ad essi correlate.
Richiesta di certificato (CR)	# ¹ tipi di certificato, tipi di certificato, # ¹ autorità certificative, autorità certificative	Utilizzato per la richiesta di certificati; specifica le tipologie di certificati richiesti e le autorità certificative accettabili.
Hash (HASH)	Dati hash	Contiene dati generati da una funzione hash.
Firma (SIG)	Dati di firma	Contiene dati generati da una funzione di firma elettronica.
Nonce (NONCE)	Dati di nonce	Contiene un nonce.
Notifica (N)	DOI, ID del protocollo, dimensione SPI, tipologia del messaggio di notifica, SPI, dati di notifica	Utilizzato per trasmettere dati di notifica quali, per es., condizioni di errore.
Cancellazione (D)	DOI, ID del protocollo, dimensione SPI, # ¹ SPI, SPI, (uno o più)	Specifica che una SA non è più valida.

¹ Il simbolo # indica "numero di".

Tabella 6.3 Tipologie di payload ISAKMP.

L'uso di più payload trasformazione consente a chi avvia la comunicazione di offrire differenti possibilità; il ricevente deve sceglierne una o rifiutare totalmente la proposta.

Il **payload trasformazione** (*transform*) definisce una trasformazione di sicurezza da utilizzare per rendere sicuro il canale di comunicazione per il protocollo designato. Il parametro transform # permette di identificare questo particolare payload in modo che il ricevente possa utilizzare tale parametro per indicare l'accettazione della trasformazione in questione. Il campo ID di trasformazione (*transform ID*) e il campo attributi (*attribute*) identificano una specifica trasformazione (per esempio, 3DES per ESP, HMAC-SHA-1-96 per AH) con gli attributi ad essa associati (per esempio, lunghezza del codice hash).

Il **payload scambio di chiavi** (*key exchange*) può essere usato per denotare una varietà di tecniche di scambio di chiavi, inclusa la tecnica Oakley, Diffie-Hellman, e la tecnica di scambio di chiavi basata su RSA utilizzata da PGP. Il campo **scambio di chiavi** contiene i dati necessari per generare una chiave di sessione. Tale campo è dipendente dallo specifico algoritmo per lo scambio di chiavi utilizzato.

Il **payload identificazione** (*identification*) è utilizzato per stabilire l'identità delle parti in comunicazione e può essere anche utilizzato per determinare l'autenticità delle informazioni. In genere, il campo **dati di identificazione** (*ID data*) conterrà un indirizzo IPv4 o IPv6.

Il **payload certificato** (*certificate*) serve per trasferire un certificato a chiave pubblica. Il campo codifica del certificato (*certificate encoding*) indica il tipo di certificato o informazioni relative al certificato stesso, tra cui:

- certificato X.509 PKCS #7
- certificato PGP
- chiave firmata DNS
- certificato X.509—firma
- certificato X.509—scambio di chiavi
- Token Kerberos
- Lista di revoca dei certificati (CRL, *certificate revocation list*)
- Lista di revoca delle autorità (ARL, *authority revocation list*)
- certificato SPKI

Nell'ambito di uno scambio ISAKMP, in ogni momento il mittente può includere un **payload richiesta di certificato** (*certificate request*) per richiedere il certificato dell'altra entità in comunicazione. Il payload può contenere più di un tipo di certificato accettabile e più di un'autorità certificativa accettabile.

Il **payload hash** contiene dati generati da una funzione hash applicata su alcune parti del messaggio e/o sullo stato ISAKMP. Questo payload può essere utilizzato per verificare l'integrità dei dati contenuti in un messaggio o per autenticare le entità coinvolte nella negoziazione.

Il **payload firma** (*signature*) contiene dati generati da una funzione di firma elettronica applicata su alcune parti del messaggio e/o sullo stato ISAKMP. Questo payload è utilizzato per verificare l'integrità dei dati contenuti in un messaggio e può essere utilizzato per servizi di non-riudio.

Il **payload nonce** contiene dati casuali utilizzati per garantire che la connessione sia attiva durante uno scambio e come protezione contro un attacco di replay.

Il **payload notifica (notification)** contiene informazioni di stato, o relative ad errori, associate alla SA corrente o alla negoziazione della SA. Sono stati definiti i seguenti messaggi ISAKMP di errore:

Tipo del payload non valido	ID del protocollo non valido	Codifica di certificato non valida
DOI non previsto	SPI non valido	Certificato non valido
Situazione non prevista	ID di trasformazione non valido	Sintassi di richiesta di certificato errata
Cookie non valido	Attributi non previsti	Autorità certificativa non valida
Versione principale non valida	Nessuna proposta scelta	Informazioni hash non valide
Versione minore non valida	Sintassi della proposta errata	Autenticazione fallita
Tipo di scambio non valido	Payload non ben formato	Firma non valida
Flag non validi	Informazioni sulle chiavi non valide	Notifica dell'indirizzo
ID del messaggio non valido		

Il solo messaggio di stato ISAKMP definito fino a ora è il messaggio *connected* (connesso). In aggiunta a tali notifiche ISAKMP, sono utilizzate specifiche notifiche DOI. Per IPSec, sono definiti i seguenti messaggi di stato aggiuntivi.

- **Tempo di validità del ricevente (responder-lifetime):** specifica il tempo di validità della SA scelta dal ricevente.
- **Status di replay (replay-status):** utilizzato dal ricevente per comunicare se effettuerà o meno i controlli per individuare attacchi di replay.
- **Contatto iniziale (initial-contact):** informa l'altra parte del fatto che si tratta della prima SA instaurata con il sistema remoto. Il ricevente di questa notifica potrà, quindi, cancellare qualunque SA che questi condivide con il sistema remoto, sotto l'ipotesi che quest'ultimo abbia effettuato un reboot e non abbia, quindi, più accesso a quelle SA.

Il **payload cancellazione (delete)** indica una o più SA che il mittente ha rimosso dalla sua base di dati e che, quindi, non sono più valide.

Scambi ISAKMP

ISAKMP fornisce un'architettura di riferimento per lo scambio di messaggi, in cui le tipologie di payload sono elementi costitutivi. La specifica prevede cinque tipi di scambio di default, riepilogati nella Tabella 6.4 in cui il termine SA denota un payload SA con i corrispondenti payload protocollo e trasformazione.

Lo **scambio base (base exchange)** permette di trasmettere congiuntamente informazioni relative allo scambio di chiavi e all'autenticazione. Questa possibilità minimizza il numero di scambi ma non protegge l'identità. I primi due messaggi forniscono i cookie e instaurano una SA con protocolli e trasformazioni convenuti; entrambe le parti utilizzano un nonce per prevenire attacchi di replay. Gli ultimi due messaggi sono usati per scambiare le chiavi, gli ID utente con il payload AUTH utilizzato per autenticare le chiavi, le identità e i nonce dei primi due messaggi.

Lo **scambio con protezione di identità (identity protection exchange)** estende lo scambio base al fine di assicurare protezione all'identità degli utenti. I primi due messaggi sono

Scambio	Note
(a) Scambio base	
(1) I → R: SA; NONCE (2) R → I: SA; NONCE (3) I → R: KE; ID _I ; AUTH (4) R → I: KE; ID _R ; AUTH	Inizio negoziazione SA-ISAKMP SA base convenuta Chiave generata; il ricevente verifica l'identità dell'iniziatore L'iniziatore verifica l'identità del ricevente; viene generata la chiave; viene instaurata la SA
(b) Scambio con protezione di identità	
(1) I → R: SA (2) R → I: SA (3) I → R: KE; NONCE (4) R → I: KE; NONCE (5)* I → R: ID _I ; AUTH (6)* R → I: ID _R ; AUTH	Inizio negoziazione SA-ISAKMP SA base convenuta Chiave generata Chiave generata Il ricevente verifica l'identità dell'iniziatore L'iniziatore verifica l'identità del ricevente; viene instaurata la SA
(c) Scambio con sola autenticazione	
(1) I → R: SA; NONCE (2) R → I: SA; NONCE; ID _R ; AUTH (3) I → R: ID _R ; AUTH	Inizio negoziazione SA-ISAKMP SA base convenuta; l'iniziatore verifica l'identità del ricevente Il ricevente verifica l'identità dell'iniziatore; viene instaurata la SA
(d) Scambio aggressivo	
(1) I → R: SA; KE; NONCE; ID _I (2) R → I: SA; KE; NONCE; ID _R ; AUTH chiave; SA base convenuta (3)* I → R: AUTH	Inizio negoziazione SA-ISAKMP e scambio di chiavi Il ricevente verifica l'identità dell'iniziatore; viene generata la chiave; SA base convenuta L'iniziatore verifica l'identità del ricevente; viene instaurata la SA
(e) Scambio informativo	
(1)* I → R: N/D	Notifica di errore o di stato, o cancellazione

Legenda:

I = iniziatore

R = ricevente

* = cifratura del payload dopo l'intestazione ISAKMP

AUTH = utilizzo del meccanismo di autenticazione.

Tabella 6.4 Tipologie di scambio ISAKMP.

utilizzati per instaurare la SA. I successivi due messaggi effettuano lo scambio delle chiavi, utilizzando nonce per la protezione contro attacchi di replay. Una volta che la chiave di sessione è stata calcolata, le due parti si scambiano messaggi cifrati che contengono informa-

zioni di autenticazione quali, ad esempio, firme digitali e, eventualmente, certificati utilizzati per accettare l'autenticità delle chiavi pubbliche.

Lo scambio con sola autenticazione (*authentication only exchange*) è utilizzato per effettuare autenticazione reciproca, senza scambio di chiavi. I primi due messaggi instaurano la SA. Inoltre, il ricevente utilizza il secondo messaggio per comunicare il proprio ID e utilizza l'autenticazione per proteggere il messaggio. Colui che ha iniziato la trasmissione invia successivamente il terzo messaggio per trasmettere il proprio ID autenticato.

Lo scambio aggressivo (*aggressive exchange*) minimizza il numero di scambi a costo di non fornire la protezione dell'identità. Nel primo messaggio, l'iniziatore propone una SA con il relativo protocollo offerto e le opzioni di trasformazione. Questi avvia anche lo scambio di chiavi e fornisce il proprio ID. Nel secondo messaggio, il ricevente indica l'accettazione della SA con un particolare protocollo e trasformazione, completa lo scambio di chiavi ed autentica le informazioni trasmesse. Nel terzo messaggio, l'iniziatore trasmette il risultato dell'autenticazione che copre le precedenti informazioni, cifrato utilizzando la chiave di sessione segreta condivisa.

Lo scambio informativo (*informational exchange*) è utilizzato per trasmettere unidirezionalmente informazioni per la gestione delle SA.

6.7 Letture consigliate e siti web

Per una trattazione più completa di IPv4 e IPv6 si rimanda il lettore a [STAL04].

In [CHEN98] viene discussa in modo dettagliato una progettazione basata su IPSec. [FRAN01] e [DORA99] forniscono un'approfondita trattazione di IPSec.

Siti web consigliati

- Progetto IPSEC di Nist: contiene articoli, presentazioni e riferimenti alle implementazioni.

6.8 Domande di revisione ed esercizi

Domande di revisione

- 1 Fornire alcuni esempi di applicazioni di IPSec.
- 2 Quali servizi sono forniti da IPSec?
- 3 Quali parametri identificano una SA e quali parametri caratterizzano la natura di una particolare SA?
- 4 Qual è la differenza tra modalità tunnel e modalità trasporto?
- 5 Che cos'è un attacco di replay?
- 6 Perché ESP comprende un campo di riempimento?

- 7 Qual è l'approccio di base per costruire delle SA?

- 8 Quali sono i ruoli del protocollo di determinazione delle chiavi di Oakley e di ISAKMP in IPSec?

Esercizi

- 1 Nell'illustrare l'elaborazione di AH, si è precisato come non tutti i campi in un'intestazione IP sono utilizzati per calcolare il MAC.

- Specificate, per ciascuno dei campi dell'intestazione IPv4, se tale campo è non modificabile, se è modificabile ma non prevedibile, oppure se è modificabile (ed è quindi posto a zero prima del computo di ICV);
- riferite le stesse considerazioni per l'intestazione IPv6;
- e per le intestazioni di estensione IPv6.

Fornite una motivazione alle risposte date per ciascuno dei casi sopra descritti e per ogni campo considerato.

- 2 Quando viene utilizzata la modalità tunnel, è generata una nuova intestazione IP esterna. Determinate, sia per IPv4 che per IPv6, le relazioni che intercorrono tra ciascun campo dell'intestazione IP esterna e ciascuna intestazione di estensione posta nel pacchetto esterno con i corrispondenti campi o intestazioni di estensione del pacchetto IP interno. Si determini, cioè, quali valori esterni sono derivati da quelli interni e quali invece sono generati in modo indipendente rispetto ai valori interni.

- 3 L'autenticazione e la cifratura end-to-end sono due caratteristiche auspicabili della comunicazione tra due host. Delineate degli schemi, simili alle Figure 6.6 e 6.9, che illustrino:
 - la transport adjacency, in cui la cifratura è applicata prima dell'autenticazione;
 - una SA in modalità trasporto all'interno di una SA in modalità tunnel, in cui la cifratura è applicata prima dell'autenticazione;
 - una SA in modalità trasporto all'interno di una SA in modalità tunnel, in cui l'autenticazione è applicata prima della cifratura.

- 4 Il documento che descrive l'architettura di IPSec stabilisce che, nel caso in cui – per consentire l'utilizzo sia del protocollo AH sia del protocollo ESP durante lo stesso flusso end-to-end – venga costituito un bundle di SA formato da due SA in modalità trasporto, l'ordine più appropriato tra l'esecuzione dei protocolli di sicurezza è effettuare il protocollo ESP prima del protocollo AH. Perché è consigliato tale approccio piuttosto che l'autenticazione prima della cifratura?

- 5 a. Quale tipologia di scambio ISAKMP (Tabella 6.4) corrisponde allo scambio aggressivo nel protocollo Oakley (Figura 6.11)?
 - Considerate lo scambio aggressivo del protocollo Oakley e determinate le corrispondenze tra i parametri contenuti in ciascun messaggio e i tipi di payload ISAKMP.

Appendice 6A Internetworking e protocolli Internet

In questa appendice viene fornita una panoramica dei protocolli Internet, presentando il ruolo del protocollo Internet nel fornire l'internetworking; in seguito saranno descritti i principali protocolli Internet, IPv4 e IPv6.

Ruolo di un protocollo Internet

Un protocollo Internet (IP) consente di interconnettere sistemi terminali localizzati su reti diverse. A tale scopo, IP è implementato sia su ciascun sistema terminale sia sui corrispondenti router, dispositivi preposti all'interconnessione di reti. I dati di più alto livello presenti in un sistema terminale sorgente sono encapsulati all'interno di un'unità dati di protocollo IP (PDU, *protocol data unit*) prima della trasmissione. La PDU attraversa quindi una o più reti e i corrispondenti router, prima di giungere all'end system di destinazione.

Il router deve essere in grado di gestire le molteplici differenze esistenti tra le diverse reti, di cui forniamo alcuni esempi.

- Schemi di indirizzamento.** Le reti possono utilizzare schemi differenti per assegnare gli indirizzi ai dispositivi coinvolti. Ad esempio, una LAN IEEE 802 utilizza come indirizzi per i dispositivi numeri binari di 16 o 48 bit; una rete pubblica X.25 a commutazione di pacchetto utilizza indirizzi decimali di 12 cifre (dove ogni cifra è codificata con 4 bit, producendo, quindi, un indirizzo di 48 bit). Inoltre, deve essere fornita una qualche forma di indirizzamento a livello globale di rete e un servizio di directory.
- Dimensioni massime di pacchetto.** A volte è necessario suddividere i pacchetti provenienti da una rete in parti più piccole per poterli trasmettere su una rete diversa. Tale operazione è nota come **frammentazione**. Ad esempio, la dimensione massima dei pacchetti su reti Ethernet è di 1500 byte, mentre solitamente reti X.25 hanno una dimensione massima dei pacchetti pari a 1000 byte. Quindi, se un pacchetto trasmesso su di una rete Ethernet deve essere trasferito su una rete X.25, sarà necessario che il corrispondente router lo frammenti in pacchetti più piccoli.
- Interfacce.** Reti diverse hanno di solito differenti interfacce hardware e software. Un router non deve essere influenzato da tali differenze.
- Affidabilità.** I servizi di rete possono variare molto rispetto all'affidabilità: in alcuni casi non viene fornita alcuna garanzia di affidabilità, mentre altri servizi possono offrire un circuito end-to-end virtuale completamente affidabile. L'operato dei router non dovrebbe basarsi quindi sull'ipotesi che la rete sia affidabile.

Le operazioni dei router dipendono dal protocollo Internet, come evidenziato nella Figura 6.13. Nell'esempio riportato, il protocollo utilizzato è il protocollo IP della suite di protocolli TCP/IP. IP deve essere implementato in tutti i sistemi terminali, le reti e i router. Inoltre, al fine di rendere possibile la comunicazione, i sistemi terminali devono avere i protocolli posti al di sopra di IP compatibili tra loro. I router intermedi necessitano di tale compatibilità soltanto fino al livello IP.

Si consideri la trasmissione di un blocco dati dal sistema terminale X a quello Y, come illustrato nella Figura 6.13. Il livello IP di X riceve da TCP i blocchi dati da inviare a Y. Il livello IP aggiunge ai dati un'intestazione che specifica l'indirizzo Internet globale di Y. L'indirizzo è composto da due parti: l'identificatore di rete e l'identificatore del sistema terminale. Il blocco così formato viene chiamato pacchetto IP. Successivamente, IP rileva che la destinazione (Y) appartenga a un'altra sottorete. Il primo passo da compiere è inviare il pacchetto a un router, nel caso specifico al router 1. Per effettuare tale operazione, IP invia i dati a LLC insieme alle informazioni di indirizzamento. LLC genera

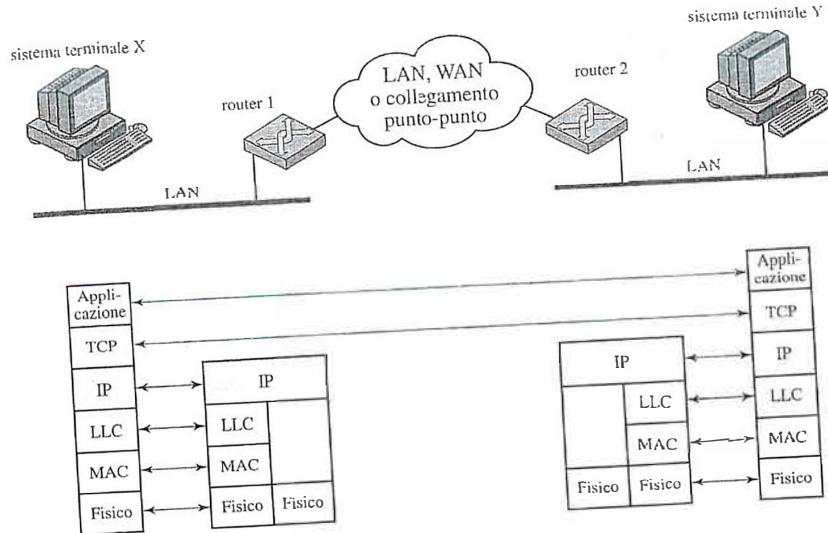


Figura 6.13 Esempio di configurazione TCP/IP.

A questo punto, il pacchetto viene trasmesso sulla LAN fino a raggiungere il router 1. Il router 1 esamina l'intestazione IP per determinare la destinazione finale dei dati, nel caso specifico Y. A questo punto il router deve prendere una decisione riguardante l'instradamento. Due sono i casi possibili:

- il sistema terminale destinazione Y è connesso in maniera diretta a una delle sottoreti cui è connesso il router;
- per raggiungere la destinazione è necessario attraversare uno o più router addizionali.

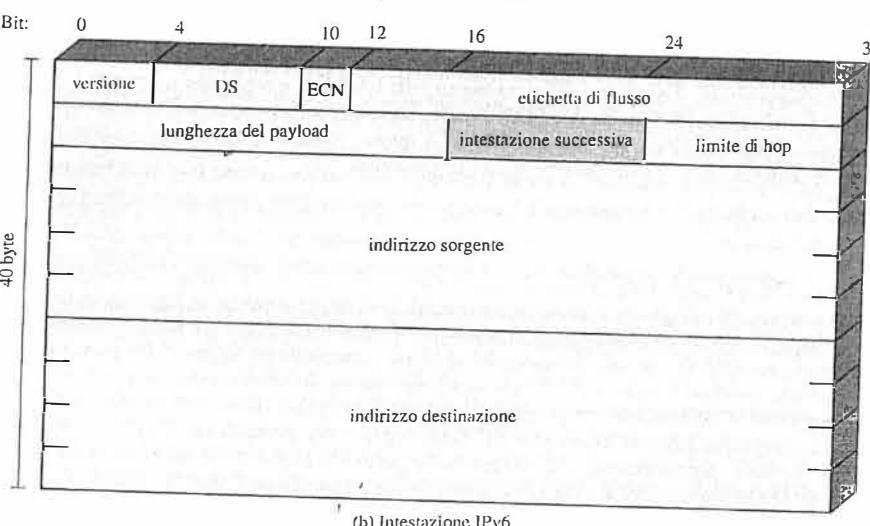
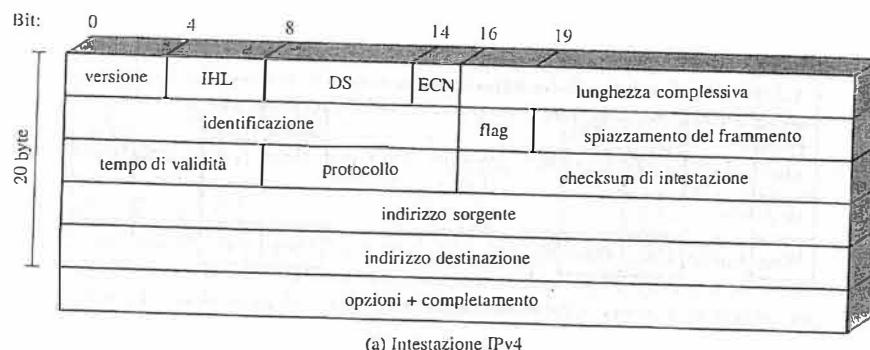
Nel nostro esempio, il pacchetto deve essere instradato al router 2 prima di raggiungere la destinazione finale. Il router 1 inoltra, quindi, il pacchetto IP al router 2 attraverso la rete intermedia, utilizzando i protocolli di tale rete. Ad esempio, se la rete intermedia è una rete X.25, l'unità dati IP è l'unico protocollo che può essere trasferito. Il pacchetto IP viene quindi inviato al router 2, dove l'intestazione viene eliminata. Il router 2 determina che il pacchetto IP è destinato a Y, che è direttamente connesso a una sottorete cui il router 2 è connesso. Successivamente, il router 2 crea un pacchetto avendo come indirizzo di destinazione quello di Y, e lo inoltra attraverso la LAN. Infine, i dati raggiungono Y, dove le intestazioni del pacchetto, LLC, le intestazioni di internet e i trailer possono essere eliminati.

Il servizio che IP offre non è affidabile. Ciò significa che IP non garantisce che tutti i dati siano recapitati o che i dati giungeranno a destinazione nell'ordine corretto. La responsabilità di riparare a tali errori è del livello successivo più alto, in questo caso TCP. Tale tipo di approccio fornisce un elevato grado di flessibilità. Dal momento che non è garantito il recapito dei dati, non viene richiesto alcun particolare requisito di affidabilità alle sottoreti coinvolte. Questo significa che il protocollo è in grado di operare con qualsiasi tipologia di sottorete. Inoltre, dato che non si garantisce il mantenimento dell'ordine dei dati alla consegna, pacchetti consecutivi possono seguire cammini differenti.

tra le reti. Questo consente al protocollo di far fronte a problemi di congestione del traffico o a guasti nella internet, semplicemente cambiando l'instradamento.

IPv4

Il fulcro dell'architettura del protocollo TCP/IP è stato per decenni il protocollo Internet (IP) versione 4 (RFC 791). La Figura 6.14a mostra il formato dell'intestazione IP, la cui dimensione minima è 20 byte, o equivalentemente 160 bit. I campi che compongono l'intestazione sono i seguenti.



DS = campo Servizi differenziati
ECN = campo Explicit congestion notification

Nota: I campi di 8 bit di DS/ECN erano precedentemente conosciuti come campo Tipo di servizio nell'intestazione IPv4 e come campo Classe di traffico nell'intestazione IPv6.

Figura 6.14 Intestazioni IP.

- **Versione, 4 bit (version):** specifica il numero di versione, per consentire l'evoluzione del protocollo; il valore di tale campo è 4.
- **Lunghezza dell'intestazione Internet, 4 bit (IHL, Internet header length):** lunghezza dell'intestazione in parole di 32 bit. Il valore minimo è cinque, corrispondente a una lunghezza minima dell'intestazione pari a 20 byte.
- **DS/ECN, (8 bit):** prima dell'introduzione dei servizi differenziati, questo campo era chiamato **Tipo di Servizio** e specificava i parametri di affidabilità, precedenza, ritardo e throughput. Questa interpretazione è stata ora superata. I primi 6 bit del campo TOS sono ora chiamati campo DS (*differentiated services*, servizi differenziati), mentre i due bit rimanenti sono riservati al campo ECN (*explicit congestion notification*).
- **Lunghezza complessiva, 16 bit (total length):** lunghezza complessiva del pacchetto IP, espressa in byte.
- **Identificazione, 16 bit (identification):** numero di sequenza che, insieme all'indirizzo sorgente, all'indirizzo destinazione e al protocollo utente, consente di identificare in modo univoco un pacchetto. Ciò significa che – per il periodo in cui il pacchetto viene trasmesso tra le reti – l'identificatore deve essere unico per l'indirizzo sorgente del pacchetto, per l'indirizzo di destinazione e per il protocollo utente.
- **Flag, 3 bit (flags):** allo stato attuale, sono stati specificati solo due dei tre bit. Quando un pacchetto viene frammentato, il bit More specifica se il frammento in questione è l'ultimo del pacchetto originario. Il bit Don't Fragment, se inizializzato, non consente la frammentazione. Questo bit può essere utile nel caso in cui si sappia che la destinazione non è in grado di riassemblare i frammenti. Per contro, l'inizializzazione di tale bit implica che il pacchetto non venga considerato qualora la sua dimensione superi la dimensione massima consentita dalle sottoreti che attraversa per raggiungere la destinazione. Nel caso in cui il bit venga inizializzato, è consigliabile utilizzare l'instradamento dalla sorgente, al fine di evitare sottoreti con limitate dimensioni massime di pacchetto.
- **Spostamento del frammento, 13 bit (fragment offset):** specifica la porzione di pacchetto originario cui corrisponde il frammento in questione. Lo spostamento è misurato in unità di 64 bit. Ciò significa che tutti i frammenti, tranne l'ultimo, devono contenere un campo dati con una dimensione multipla di 64 bit.
- **Tempo di validità, 8 bit (TTL, time to live):** specifica per quanto tempo, in secondi, un pacchetto può rimanere su Internet. Ogni router che il pacchetto attraversa deve decrementare il TTL di almeno uno; quindi, il TTL è simile a un conteggio dei salti.
- **Protocollo, 8 bit (protocol):** specifica il protocollo di livello immediatamente superiore, che deve ricevere il campo dati alla destinazione; tale campo identifica, quindi, il tipo dell'intestazione successiva all'intestazione IP nel pacchetto.
- **Somma di controllo di intestazione, 16 bit (header checksum):** codice per la rivelazione di errori che viene applicato esclusivamente all'intestazione. Dal momento che alcuni campi dell'intestazione possono essere modificati durante la trasmissione (per esempio, i campi tempo di validità e quelli relativi alla segmentazione), tale codice viene verificato e ricalcolato da ogni router. Il campo somma di controllo è ottenuto effettuando l'addizione in complemento a uno di tutte le parole di 16 bit presenti nell'intestazione. Per effettuare tale computazione, il campo somma di controllo è inizializzato a zero.
- **Indirizzo sorgente, 32 bit (source address):** codificato in modo da consentire un'assegnazione variabile del numero di bit che devono rappresentare la rete e il sistema terminale connesso alla rete in questione (7 e 24 bit, 14 e 16 bit o 21 e 8 bit).

- **Indirizzo destinazione, 32 bit (destination address):** ha le stesse caratteristiche dell'indirizzo sorgente.
- **Opzioni, lunghezza variabile (options):** contiene le opzioni richieste dall'utente che trasmette il pacchetto; tali opzioni possono includere un'etichetta di sicurezza, l'instradamento dalla sorgente, la registrazione della sorgente e un timestamp.
- **Completamento, lunghezza variabile (padding):** utilizzato per garantire che la lunghezza dell'intestazione del pacchetto sia un multiplo di 32 bit.

IPv6

Nel 1995, l'Internet Engineering Task Force (IETF), che sviluppa i protocolli standard per Internet, pubblicò la specifica di un protocollo Internet di nuova generazione, che divenne noto come IPng (RFC 1752). Nel 1996, questa specifica è stata convertita in uno standard noto come IPv6. IPv6 fornisce migliori funzionalità rispetto al protocollo Internet esistente (cioè IPv4), dal momento che è stato progettato per gestire le più elevate velocità delle reti odiere e flussi di dati multimediali, inclusi grafica e video, che stanno diventando sempre più preponderanti. Ma il maggior impulso allo sviluppo di un nuovo protocollo è legato alla necessità di uno spazio di indirizzamento più ampio. In seguito al rapido sviluppo di Internet e delle reti private ad essa connesse, gli indirizzi a 32 bit – utilizzati da IPv4 per specificare sia la sorgente sia la destinazione – risultano insufficienti per soddisfare tutte le richieste di nuovi indirizzi. IPv6 prevede campi di 128 bit per l'indirizzo sorgente e destinazione, come mostrato nella Figura 6.14b. È facilmente prevedibile che tutte le installazioni che utilizzano TCP/IP migreranno dalla versione di IP utilizzata correntemente a IPv6, ma tale migrazione richiederà molti anni, se non addirittura decenni.

Intestazione IPv6

L'intestazione IPv6 ha una lunghezza fissa pari a 40 byte, e contiene i seguenti campi (Figura 6.14b).

- **Versione, 4 bit (version):** numero di versione del protocollo Internet; il valore di tale campo è 6.
- **DS/ECN, 8 bit:** prima dell'introduzione dei servizi differenziati, questo campo era chiamato **Classe di traffico** ed era riservato per essere usato dal nodo di origine e/o dai router, che inoltravano il traffico, per identificare e per distinguere tra pacchetti IPv6 di differenti classi o priorità. I primi 6 bit del campo Classe di Traffico sono ora chiamati campo DS (*differentiated service, servizi differenziati*), mentre i due bit rimanenti sono riservati per un campo ECN (*explicit congestion notification*).
- **Etichetta di flusso, 20 bit (flow label):** può essere utilizzata da un host per etichettare quei pacchetti che richiedono una gestione particolare da parte dei router. Tale tipo di etichettamento può essere di aiuto nell'allocatione delle risorse e nella gestione del traffico in tempo reale.
- **Lunghezza del payload, 16 bit (payload length):** lunghezza della parte di pacchetto IPv6 che segue l'intestazione, espressa in byte. In altri termini, tale campo indica la lunghezza complessiva di tutte le intestazioni di estensione e della PDU di livello trasporto.
- **Intestazione successiva, 8 bit (next header):** identifica il tipo dell'intestazione immediatamente successiva all'intestazione IPv6; tale intestazione può essere sia un'intestazione di estensione IPv6 sia un'intestazione di livello più alto come, ad esempio, TCP o UDP.
- **Limite di hop, 8 bit (hop limit):** numero massimo di hop che il pacchetto in questione può ancora effettuare. Tale campo è inizializzato dalla sorgente al valore desiderato e decrementato di un'unità da ogni nodo che inoltra il pacchetto. Il pacchetto è scartato quanto tale campo assume il valore zero.

- **Indirizzo sorgente, 128 bit (source address):** indirizzo della sorgente del pacchetto.
- **Indirizzo destinazione, 128 bit (destination address):** indirizzo di colui che dovrebbe ricevere il pacchetto. Tale indirizzo può non essere quello del destinatario finale nel caso in cui sia presente un'intestazione di estensione di instradamento (*routing*), come verrà spiegato più in dettaglio in seguito.

Sebbene l'intestazione IPv6 sia più lunga della porzione obbligatoria dell'intestazione IPv4 (40 byte contro 20), essa contiene meno campi (8 contro 12). I router devono, quindi, effettuare meno elaborazione per ciascuna intestazione, il che dovrebbe incrementare la velocità di instradamento.

Intestazioni di estensione IPv6

Un pacchetto IPv6 contiene l'intestazione appena descritta e zero o più intestazioni di estensione. Al di fuori di IPSec, sono state definite le seguenti intestazioni di estensione.

- **Intestazione con opzioni hop-by-hop (hop-by-hop options header):** definisce particolari opzioni che richiedono un'elaborazione hop-by-hop (cioè devono essere esaminate da tutti i router sul percorso).
- **Intestazione di instradamento (routing header):** fornisce modalità di instradamento esteso, simile all'instradamento dalla sorgente di IPv4.
- **Intestazione di frammentazione (fragment header):** contiene informazioni relative alla frammentazione e al riassemblaggio.
- **Intestazione di autenticazione (authentication header):** fornisce integrità e autenticazione del pacchetto.
- **Intestazione ESP (encapsulating security payload header):** fornisce riservatezza.
- **Intestazione con opzioni per la destinazione (destination options header):** contiene informazioni optionali che sono esaminate dal nodo destinazione.

Lo standard IPv6 raccomanda che, nel caso vengano utilizzati molteplici intestazioni di estensione, le intestazioni IPv6 appaiano nel seguente ordine.

1. Intestazione IPv6: obbligatoria, deve sempre apparire per prima.
2. Intestazione con opzioni hop-by-hop.
3. Intestazione con opzioni per la destinazione: relativa a opzioni che devono essere considerate sia dalla prima destinazione che appare nel campo indirizzo destinazione sia dalle destinazioni successive che compaiono nell'intestazione di instradamento.
4. Intestazione di instradamento.
5. Intestazione di frammentazione.
6. Intestazione di autenticazione.
7. Intestazione ESP.
8. Intestazione con opzioni per la destinazione: relativa a opzioni che devono essere considerate solo dalla destinazione finale del pacchetto.

La Figura 6.15 illustra un esempio di pacchetto IPv6 che contiene un caso di ciascuna delle intestazioni sopra descritte, ad eccezione di quelle che contengono informazioni relative alla sicurezza. Si noti che sia l'intestazione IPv6 sia le intestazioni di estensione contengono il campo intestazione successiva che descrive il tipo dell'intestazione immediatamente successiva. Se l'intestazione suc-

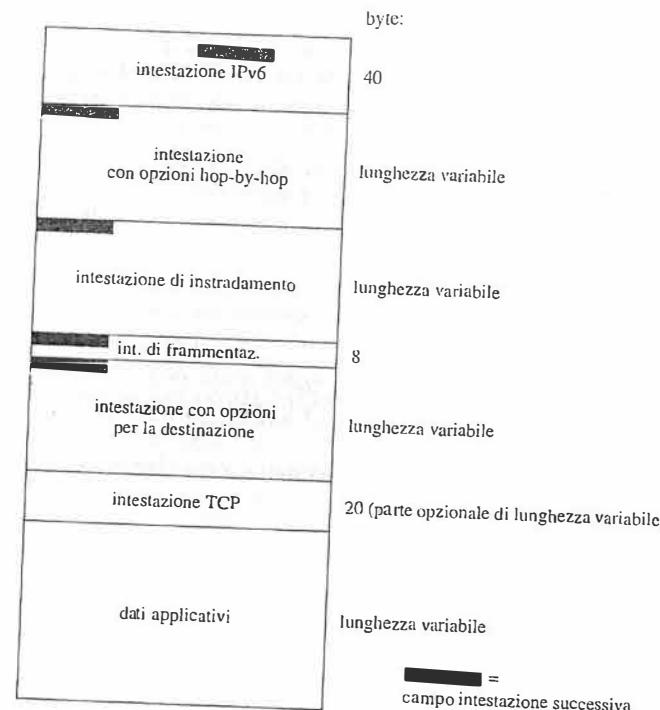


Figura 6.15 Pacchetto IPv6 con intestazioni di estensione (contenente un segmento TCP).

cessiva è un'intestazione di estensione, questo campo contiene l'identificatore del tipo dell'intestazione. In caso contrario, tale campo contiene l'identificatore del protocollo di livello superiore che utilizza IPv6 (solitamente un protocollo di livello trasporto), utilizzando gli stessi valori usati per il campo protocollo di IPv4. Nella figura, il protocollo di livello superiore è TCP, e quindi i dati di livello superiore contenuti nel pacchetto IPv6 consistono di un'intestazione TCP seguita da un blocco di dati applicativi.

L'intestazione con opzioni hop-by-hop fornisce informazioni opzionali che, se presenti, devono essere prese in considerazione da ciascun router lungo il percorso. L'intestazione consiste nei seguenti campi:

- Intestazione successiva, 8 bit (*next header*):** specifica il tipo dell'intestazione che segue immediatamente l'intestazione in questione.
- Lunghezza dell'intestazione di estensione, 8 bit (*header extension length*):** lunghezza dell'intestazione in unità di 64 bit, ad esclusione dei primi 64 bit.
- Opzioni (*options*):** contiene una o più opzioni. Ogni opzione è formata da tre sotto-campi: un'etichetta che indica il tipo di opzione, una lunghezza e un valore.

Per il momento è stata definita la sola opzione jumbo payload, utilizzata per trasmettere pacchetti IPv6 aventi payload con una lunghezza superiore a $2^{16} - 1 = 65.535$ byte. Il campo dati di questa op-

zione è lungo 32 bit e contiene la lunghezza del pacchetto espressa in byte, escludendo l'intestazione IPv6. Per tali pacchetti, il campo lunghezza del payload nell'intestazione IPv6 deve essere posto a zero, e non deve essere presente alcuna intestazione di frammentazione. Tramite tale opzione, IPv6 fornisce supporto per dimensioni di pacchetto fino a 4 miliardi di byte. Ciò semplifica la trasmissione di grossi pacchetti contenenti video e fa sì che IPv6 possa utilizzare al meglio la capacità di trasmissione disponibile su ogni mezzo trasmissivo utilizzato.

L'intestazione di instradamento elenca uno o più nodi intermedi che devono essere attraversati per raggiungere la destinazione del pacchetto. Le intestazioni di instradamento iniziano sempre con un blocco di 32 bit che contiene quattro campi di 8 bit. Tale blocco è seguito da dati di instradamento che variano a seconda del tipo di instradamento considerato. I quattro campi da 8 bit sono i seguenti.

- Intestazione successiva (*next header*):**
- Lunghezza dell'intestazione di estensione (*header extension length*):**
- Tipologia di instradamento (*routing type*):** specifica una particolare variante dell'intestazione di instradamento. Nel caso non riconosca il valore di questo campo, il router deve scartare il pacchetto.
- Segmenti rimanenti (*segment left*):** numero di nodi intermedi esplicitamente elencati che devono essere ancora attraversati prima di raggiungere la destinazione finale.

Oltre a questa definizione di carattere generale dell'intestazione, la specifica IPv6 definisce l'intestazione di instradamento di tipo 0. Quando viene utilizzato questo tipo di intestazione, il nodo sorgente non include l'indirizzo della destinazione finale nell'intestazione IPv6. Al contrario, tale indirizzo è l'ultimo indirizzo elencato nell'intestazione di instradamento, mentre l'intestazione IPv6 contiene l'indirizzo del primo router che si vorrebbe incontrare sul percorso. L'intestazione di instradamento è esaminata solo quando il pacchetto raggiunge il nodo corrispondente all'indirizzo nell'intestazione IPv6. A questo punto, il contenuto dell'intestazione IPv6 e dell'intestazione di instradamento vengono aggiornati e il pacchetto è inoltrato. L'aggiornamento consiste nel mettere l'indirizzo successivo da visitare nell'intestazione IPv6 e nel decrementare il campo segmenti rimanenti nell'intestazione di instradamento.

IPv6 stabilisce che un nodo IPv6 inverta i cammini nei pacchetti ricevuti che contengono un'intestazione di instradamento, per restituire un pacchetto al mittente.

L'intestazione di frammentazione è utilizzata da una sorgente nel caso in cui sia necessaria la frammentazione. In IPv6, la frammentazione può essere effettuata solo dai nodi sorgente, e non dai router posti sul cammino che un pacchetto deve compiere per raggiungere la sua destinazione finale. Per sfruttare al meglio l'ambiente di internetworking, un nodo deve eseguire un algoritmo di individuazione del cammino con cui determina la più piccola unità di trasmissione massima (MTU) di ogni sottorete appartenente al cammino. In altri termini, l'algoritmo di individuazione del cammino consente a un nodo di conoscere la dimensione della MTU della rete che costituisce un "collo di bottiglia" lungo il cammino. Avendo questa informazione, il nodo sorgente frammenterà in modo corretto il pacchetto, per ogni indirizzo destinazione considerato. Se il nodo sorgente non fosse a conoscenza di tale informazione dovrebbe limitare la dimensione di tutti i pacchetti a 1280 byte, che è la dimensione minima di una MTU, per una qualsiasi sottorete.

In aggiunta al campo intestazione successiva, l'intestazione di frammentazione contiene i seguenti campi:

- Spostamento del frammento, 13 bit (*fragment offset*):** specifica la porzione di pacchetto originario cui corrisponde il payload del frammento in questione. Tale spostamento è misurato in unità di 64 bit. Questo implica che tutti i frammenti (ad eccezione dell'ultimo) devono contenere un campo dati con dimensione multipla di 64 bit.

- Res, 2 bit: riservato per utilizzi futuri.
- Flag M, 1 bit (*M flag*): se tale flag è posto a 1 significa che il frammento in questione non è l'ultimo, se è posto a 0 indica che si tratta dell'ultimo frammento.
- Identificazione, 32 bit (*identification*): utilizzato per identificare in modo univoco il pacchetto originario. L'identificatore deve essere unico per l'indirizzo sorgente e destinazione del pacchetto per il tempo in cui il pacchetto rimane nella internet. Tutti i frammenti aventi lo stesso identificatore e lo stesso indirizzo sorgente e destinazione sono riassemblati per ricostituire il pacchetto originario.

L'intestazione con opzioni per la destinazione fornisce informazioni opzionali che, se presenti, sono prese in considerazione solo dal nodo destinazione del pacchetto. Tale intestazione ha lo stesso formato dell'intestazione con opzioni hop-by-hop.

Capitolo 7

Sicurezza web

Attualmente il numero di persone, aziende ed enti pubblici presenti sul web con un proprio sito, o semplicemente collegati a Internet, è in costante, rapida espansione. Sviluppo che coinvolge anche le attività legate al commercio elettronico. Per contro, Internet e il Web risultano estremamente vulnerabili ad attacchi di varia natura; cresce quindi la domanda di servizi web sicuri.

L'argomento della sicurezza web è così ampio che servirebbe un intero libro per coprirlo. In questo capitolo tratteremo in primo luogo i requisiti generali della sicurezza web per concentrarci poi su due schemi standard che stanno assumendo crescente importanza come componenti del commercio sul Web: SSL/TLS e SET.

7.1 Considerazioni sulla sicurezza web

Il World Wide Web è fondamentalmente un'applicazione client/server che gira su Internet e sulle internet TCP/IP. Pertanto, gli strumenti e gli approcci finora descritti nel libro sono rilevanti per la sicurezza web. Tuttavia, come evidenziato in [GARF97], il Web presenta nuove problematiche rispetto all'ambito della sicurezza dei sistemi di elaborazione e della sicurezza di rete.

- Internet è uno strumento bidirezionale. A differenza dei sistemi di comunicazione tradizionali o di quelli elettronici – che coinvolgono teletext, risposta vocale o via fax – il Web è vulnerabile ad attacchi diretti ai server web presenti su Internet.
- Il Web sta sempre più assumendo il ruolo di mercato ad alta visibilità per informazioni aziendali e di prodotto, e il ruolo di piattaforma per transazioni commerciali. Un sovvertimento dei server web potrebbe, quindi, danneggiare gravemente la reputazione delle aziende presenti sulla rete e causare ingenti perdite economiche.
- I server web sono molto semplici da utilizzare e relativamente facili da configurare e gestire; inoltre, è sempre più facile sviluppare contenuti web. Tuttavia, il software sottostante è straordinariamente complesso; può quindi presentare parecchi punti deboli dal punto di vista della sicurezza. La breve storia del Web è piena di esempi di nuovi sistemi e di aggiornamenti, opportunamente installati, che risultano vulnerabili a una serie di attacchi alla sicurezza.

- Un server web può essere sfruttato come rampa di lancio verso il complesso dei sistemi di elaborazione delle aziende o delle organizzazioni. Una volta sovvertito un server web, un avversario può ottenere l'accesso a dati e sistemi che non fanno parte del Web ma che sono connessi al server nel sito locale.
- I comuni utenti dei servizi web non hanno, ovviamente, una specifica formazione in tema di sicurezza. Non sono, quindi, informati sui rischi di sicurezza esistenti né possiedono gli strumenti o la conoscenza per attuare le necessarie contromisure.

Minacce alla sicurezza web

La Tabella 7.1 riporta una sintesi delle tipologie di minacce alla sicurezza cui si è esposti quando si utilizza il Web.

Una prima modalità in base alla quale classificare le minacce alla sicurezza di un sito web è distinguere in attacchi passivi e attivi. Gli attacchi passivi comprendono: l'intercettazione del traffico di rete fra browser e server e la capacità di ottenere informazioni contenute in un sito web che dovrebbe essere ad accesso limitato. Gli attacchi attivi includono: l'assunzione dell'identità di un altro utente, l'alterazione dei messaggi in transito fra un client e un server e la modifica delle informazioni presenti su un sito web.

Una seconda modalità in base alla quale classificare le minacce alla sicurezza di un sito web consiste nella localizzazione della minaccia: server web, browser web e traffico di rete fra browser e server. Le problematiche di sicurezza relative a server e browser ricadono nella categoria della sicurezza dei sistemi di elaborazione. Questo argomento verrà affrontato nella Parte Quarta del libro, insieme a più generali problematiche di sicurezza di sistema. Le problematiche relative alla sicurezza del traffico di rete ricadono nella categoria della sicurezza di rete e sono considerate in questo capitolo.

Approcci alla sicurezza del traffico web

Per fornire sicurezza web, sono possibili una serie di approcci. I diversi approcci considerati sono simili dal punto di vista dei servizi offerti e, in un certo senso, nei meccanismi impiegati, ma differiscono per quanto concerne l'ambito di applicabilità e il loro posizionamento all'interno della pila del protocollo TCP/IP.

Questa differenza è mostrata nella Figura 7.1. Una modalità per fornire sicurezza web consiste nell'utilizzare la sicurezza IP (Figura 7.1a). IPSec ha il vantaggio di essere traspa-

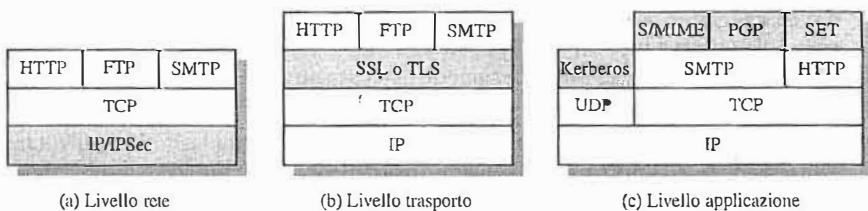


Figura 7.1 Posizionamento dei meccanismi di sicurezza all'interno della pila del protocollo TCP/IP.

Minacce	Conseguenze	Contromisure
Integrità	<ul style="list-style-type: none"> • Modifica dei dati utente • Browser con cavallo di Troia • Modifica della memoria • Modifica del traffico di messaggi in transito 	<ul style="list-style-type: none"> • Somma di controllo critografica
Riservatezza	<ul style="list-style-type: none"> • Intercettazione sulla rete • Furto di informazioni dal server • Furto di dati dal client • Acquisizione di informazioni sulla configurazione di rete • Acquisizione di informazioni su quale client dialoga con il server 	<ul style="list-style-type: none"> • Cifratura, proxy web
Negazione di servizio	<ul style="list-style-type: none"> • Eliminazione dei thread utente • Temporeggiare la macchina con false richieste • Saturazione del disco o della memoria • Isolamento della macchina tramite attacchi DNS 	<ul style="list-style-type: none"> • Conseguenze distruttive • Grave fastidio • Impossibilità per l'utente di eseguire il proprio lavoro
Autenticazione	<ul style="list-style-type: none"> • Assunzione dell'identità di utenti legittimi • Falsificazione dei dati 	<ul style="list-style-type: none"> • Errata identificazione degli utenti • Fiducia verso informazioni false

Tabella 7.1 Sintesi delle tipologie di minacce web [RUB197].

rente agli utenti finali e alle applicazioni e di fornire una soluzione di uso generale. Inoltre, IPsec comprende una funzionalità di filtro in modo che solamente una parte del traffico debba essere sottoposto all'elaborazione aggiuntiva IPsec.

Una seconda soluzione relativamente di uso generale consiste nell'implementare la sicurezza appena sopra TCP (Figura 7.1b). Il primo esempio di questo tipo di approccio è rappresentato dal **secure sockets layer** (SSL, livello delle socket sicure) e dal successivo standard Internet conosciuto come **transport layer security** (TLS, sicurezza di livello trasporto). A questo livello si presentano due scelte di implementazione. Per maggiore generalità, SSL (o TLS) potrebbe essere fornito come parte della suite di protocolli sottostante e, quindi, risultare trasparente alle applicazioni. Alternativamente, si può incorporare SSL all'interno di specifici pacchetti software. Ad esempio, i browser Netscape e Microsoft Explorer sono dotati di SSL, e la maggior parte dei server web ha implementato questo protocollo.

I servizi di sicurezza specificamente connessi a un'applicazione sono incorporati all'interno dell'applicazione stessa. La Figura 7.1c mostra esempi di questa architettura. Il vantaggio è che il servizio può essere personalizzato in base alle specifiche esigenze di una certa applicazione.

Nell'ambito della sicurezza web, un esempio importante di tale approccio è dato dal meccanismo **secure electronic transaction** (SET, transazioni elettroniche sicure)¹.

La rimanente parte del capitolo è dedicata alla discussione di SSL/TLS e di SET.

7.2 Secure socket layer (SSL) e transport layer security (TLS)

SSL è stato creato da Netscape. La versione 3 del protocollo è stata progettata con una revisione pubblica e con informazioni di provenienza aziendale ed è stata pubblicata come documento Internet draft. Successivamente, una volta raggiunto il consenso a sottomettere il protocollo al processo di standardizzazione Internet, all'interno di IETF è stato costituito il gruppo di lavoro TLS per lo sviluppo di uno standard comune. Si può considerare la prima versione di TLS essenzialmente come un SSLv3.1, molto vicino e compatibile con SSLv3.

Il fulcro di questo paragrafo concerne la presentazione di SSLv3 e si conclude con la descrizione delle differenze fondamentali fra SSLv3 e TLS.

Architettura SSL

SSL è progettato per far uso del protocollo TCP al fine di fornire un servizio di sicurezza end-to-end affidabile. Come mostra la Figura 7.2, SSL non è un unico protocollo, ma è piuttosto costituito da due livelli di protocolli.

Il protocollo SSL Record fornisce servizi di sicurezza di base a un certo numero di protocolli di più alto livello. In particolare, il protocollo HTTP (*hypertext transfer protocol*) che

¹ La Figura 7.1c mostra SET sopra HTTP; questa è un'implementazione comune. In altre implementazioni SET fa direttamente uso di TCP.

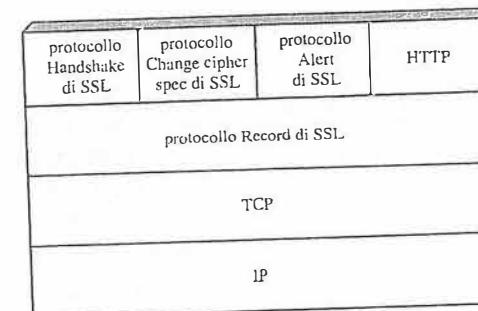


Figura 7.2 Pila del protocollo SSL.

fornisce il servizio di trasferimento per le interazioni client/server web, può operare sopra SSL. SSL contiene tre protocolli di più alto livello: il protocollo Handshake, il protocollo Change cipher spec e il protocollo Alert. Questi protocolli specifici di SSL vengono utilizzati nella gestione degli scambi SSL e sono presi in esame più avanti in questo paragrafo.

Due concetti importanti di SSL sono quello di connessione SSL e di sessione.

- **Connessione.** Una connessione è un trasporto (nella definizione del modello a strati OSI) che fornisce una tipologia opportuna di servizio. Per SSL, queste connessioni sono relazioni tra entità paritarie (*peer-to-peer*). Le connessioni sono transitorie. Ogni connessione è associata a una sessione.
- **Sessione.** Una sessione SSL è un'associazione fra un client e un server. Le sessioni sono create dal protocollo Handshake e definiscono un insieme di parametri crittografici di sicurezza, che possono essere condivisi fra molteplici connessioni. Le sessioni si utilizzano per evitare una costosa negoziazione di nuovi parametri di sicurezza per ciascuna connessione.

Ci possono essere molteplici connessioni sicure fra una qualunque coppia di parti (applicazioni come HTTP su client e server). In teoria ci potrebbero essere anche molteplici sessioni simultanee fra due parti, ma questa funzionalità non è utilizzata nella pratica.

A ciascuna sessione è associato un certo numero di stati. Una volta instaurata una sessione, c'è uno stato operativo corrente sia per la lettura sia per la scrittura (per esempio, ricezione e trasmissione). Inoltre, durante il protocollo Handshake vengono creati gli stati di attesa (*pending*) per lettura e scrittura. Alla positiva conclusione del protocollo Handshake, gli stati di attesa diventano stati correnti.

Lo stato di una sessione è definito dai seguenti parametri (le cui definizioni sono tratte dalla specifica SSL).

- **Identificatore di sessione** (*session identifier*): sequenza arbitraria di byte scelta dal server per identificare uno stato di sessione attivo o riattivabile.
- **Certificato dell'entità prioritaria** (*peer certificate*): certificato X509.v3 dell'entità prioritaria. Questo elemento dello stato può essere nullo.

- Metodo di compressione (compression method)**: algoritmo usato per la compressione dei dati prima della cifratura.
- Specifiche del cifrario (cipher spec)**: specifica l'algoritmo di cifratura dei dati che viene utilizzato – come, ad esempio, nessun algoritmo (null), AES, etc. – e l'algoritmo hash (ad esempio, MD5 o SHA-1) utilizzato per il calcolo del MAC. Definisce anche gli attributi crittografici come, ad esempio, la dimensione del codice hash (hash_size).
- Valore segreto principale (master secret)**: valore segreto a 48 byte condiviso fra il client e il server.
- È riattivabile (is resumable)**: flag che indica se la sessione può essere usata per instaurare nuove connessioni.

Uno stato di connessione è definito dai seguenti parametri.

- Valore casuale di server e client (server and client random)**: sequenze di byte scelte dal server e dal client per ciascuna connessione.
- MAC segreto del server per la scrittura (server write MAC secret)**: chiave segreta utilizzata nelle operazioni MAC sui dati inviati dal server.
- MAC segreto del client per la scrittura (client write MAC secret)**: chiave segreta utilizzata nelle operazioni MAC sui dati inviati dal client.
- Chiave di scrittura del server (server write key)**: chiave di cifratura convenzionale per i dati cifrati dal server e decifrati dal client.
- Chiave di scrittura del client (client write key)**: chiave di cifratura convenzionale per i dati cifrati dal client e decifrati dal server.
- Vettori di inizializzazione (initialization vectors)**: quando si utilizza un cifrario a blocchi in modalità CBC, viene mantenuto un vettore di inizializzazione (IV) per ciascuna chiave. Tale campo è dapprima inizializzato dal protocollo Handshake di SSL. Successivamente, l'ultimo blocco di testo cifrato di ciascun record viene mantenuto per essere utilizzato come IV per il record successivo.
- Numeri di sequenza (sequence numbers)**: ciascuna parte mantiene, per ogni connessione, numeri di sequenza separati per i messaggi trasmessi e ricevuti. Quando una parte invia o riceve un messaggio change cipher spec per la modifica della specifica del cifrario, si pone a zero il numero di sequenza appropriato. I numeri di sequenza non possono superare il valore $2^{64} - 1$.

Protocollo Record di SSL

Il protocollo Record di SSL fornisce due servizi per le connessioni SSL, grazie al protocollo Handshake.

- Riservatezza**: il protocollo Handshake definisce una chiave segreta condivisa, che viene usata per la cifratura convenzionale del payload SSL.
- Integrità dei messaggi**: il protocollo Handshake definisce anche una chiave segreta, che viene usata per creare il codice di autenticazione del messaggio (MAC).

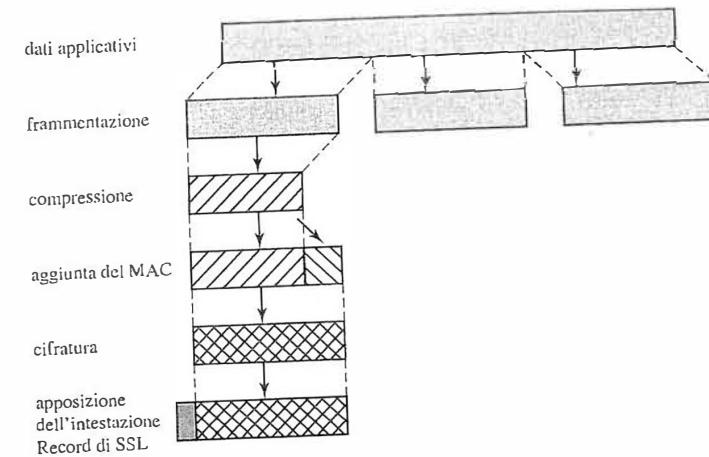


Figura 7.3 Funzionamento del protocollo Record di SSL.

La Figura 7.3 riporta il funzionamento complessivo del protocollo Record di SSL. Il protocollo Record prende un messaggio applicativo da trasmettere, effettua la frammentazione dei dati in blocchi più maneggevoli, eventualmente esegue la compressione dei dati, applica il MAC, esegue la cifratura, aggiunge un'intestazione e trasmette il messaggio risultante in un segmento TCP. La chiave segreta condivisa utilizzata per la cifratura convenzionale dei payload di SSL e quella utilizzata per creare il codice di autenticazione del messaggio (MAC) sono definite tramite il protocollo Handshake. I dati ricevuti sono decifrati, verificati, decompressi, riassemblati e infine consegnati agli utenti a livello più alto.

Il primo passo è la **frammentazione**. Ogni messaggio dei livelli sovrastanti viene frammentato in blocchi di dimensione 2^{14} byte (16384 byte) o di dimensione inferiore. Successivamente e facoltativamente, si applica la **compressione**. La compressione deve essere senza perdita e non dovrebbe aumentare la lunghezza del contenuto di più di 1024 byte². In SSLv3 (come anche nella versione corrente di TLS), non è specificato alcun algoritmo di compressione; quindi, il valore di default per l'algoritmo di compressione è null.

Il passo di elaborazione successivo consiste nel calcolare un MAC sui dati compressi. A tale scopo viene utilizzata una chiave segreta condivisa. Il calcolo del MAC è definito come segue:

```

hash(MAC_write_secret || pad_2 ||
hash(MAC_write_secret || pad_1 || seq_num ||
SSLCompressed.type ||
SSLCompressed.length || SSLCompressed.fragment))

```

² Ovviamente, si spera che la compressione riduca e non espanda i dati. Tuttavia, per blocchi molto piccoli, può verificarsi quest'ultima eventualità a causa delle convenzioni di formattazione.

dove

	= operatore di concatenazione
MAC_write_secret	= chiave segreta condivisa
hash	= algoritmo hash crittografico (MD5 o SHA-1)
pad_1	= il byte 0x36 (0011 0110) ripetuto 48 volte (384 bit) per MD5 e 40 volte (320 bit) per SHA-1
pad_2	= il byte 0x5C (0101 1100) ripetuto 48 volte (384 bit) per MD5 e 40 volte (320 bit) per SHA-1
seq_num	= numero di sequenza per il messaggio considerato
SSLCompressed.type	= protocollo di livello più alto utilizzato per elaborare il frammento considerato
SSLCompressed.length	= lunghezza del frammento compresso
SSLCompressed.fragment	= frammento compresso (se non si utilizza la compressione, il frammento è in chiaro)

Come si può notare, l'algoritmo appena descritto è molto simile all'algoritmo HMAC definito nel Capitolo 3. La differenza è che in SSLv3 i due pad sono concatenati mentre in HMAC sono combinati mediante OR esclusivo. L'algoritmo MAC di SSLv3 è basato sull'Internet draft originale di HMAC, in cui si usava la concatenazione. La versione finale di HMAC, definita nel documento RFC 2104, usa l'OR esclusivo.

Il messaggio compresso e il MAC sono quindi **cifrati** attraverso cifratura simmetrica. La cifratura non dovrebbe aumentare la lunghezza del contenuto di più di 1024 byte, in modo tale che la lunghezza totale non dovrebbe essere superiore a $2^{14} + 2048$. Gli algoritmi di cifratura consentiti sono i seguenti:

Cifratura a blocchi		Cifratura a flusso	
Algoritmo	Dimensioni della chiave	Algoritmo	Dimensioni della chiave
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza può essere usato negli schemi di cifratura con smart-card.

Per la cifratura a flusso, vengono cifrati il messaggio compresso e il MAC associato. Si noti che il MAC è calcolato prima di eseguire la cifratura e, quindi, viene cifrato insieme al testo in chiaro o con il testo in chiaro compresso.

Per la cifratura a blocchi, il completamento può essere aggiunto dopo il calcolo del MAC e prima della cifratura. Il completamento consiste in un certo numero di byte aggiuntivi se-

guiti da un byte che specifica la lunghezza del completamento stesso. La quantità totale del completamento corrisponde alla più piccola quantità tale che la dimensione totale dei dati da cifrare (testo in chiaro più MAC più completamento) sia un multiplo della lunghezza del blocco del cifrario. Un esempio potrebbe essere un testo in chiaro (o compresso) di 58 byte con un MAC di 20 byte (impiegando SHA-1) cifrati utilizzando una lunghezza di blocco di 8 byte (per esempio DES). Contando anche il byte padding.length, si arriva a un totale di 79 byte. Per rendere il valore totale un intero multiplo di 8, si aggiunge un byte di completamento.

L'ultimo passo dell'elaborazione del protocollo Record di SSL consiste nell'apporre un'intestazione composta dai seguenti campi:

- **Tipo di contenuto, 8 bit (content type):** il protocollo di livello più alto utilizzato per elaborare il frammento accluso.
- **Versione principale, 8 bit (major version):** indica la versione principale di SSL in uso. Nel caso di SSLv3, il valore è 3.
- **Versione minore, 8 bit (minor version):** indica la versione minore di SSL in uso. Nel caso di SSLv3, il valore è 0.
- **Lunghezza compressa, 16 bit (compressed length):** lunghezza in byte del frammento in chiaro (o del frammento compresso se si usa la compressione). Il valore massimo è $2^{14} + 2048$.

I tipi di contenuto definiti sono change_cipher_spec, alert, handshake, e application_data. I primi tre sono i protocolli specifici di SSL, descritti in seguito. Si noti che non vi è distinzione fra le diverse applicazioni che possono utilizzare SSL (per esempio, HTTP); il contenuto dei dati generati da queste applicazioni non è noto a SSL.

La Figura 7.4 mostra il formato generato dal protocollo Record di SSL.

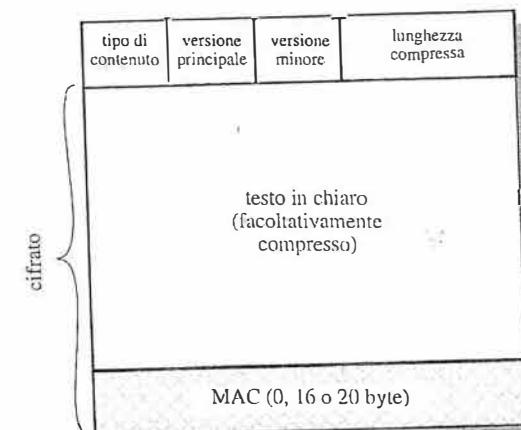


Figura 7.4 Formato record di SSL.

Protocollo Change cipher spec

Il protocollo Change cipher spec è il più semplice fra i tre specifici protocolli di SSL che utilizzano il protocollo Record di SSL. Consiste in un singolo messaggio (Figura 7.5a), che a sua volta consiste in un solo byte con valore 1. L'unico scopo di questo messaggio è quello di forzare la copiatura dello stato di attesa nello stato corrente, aggiornando la suite di cifratura da usare nella connessione corrente.

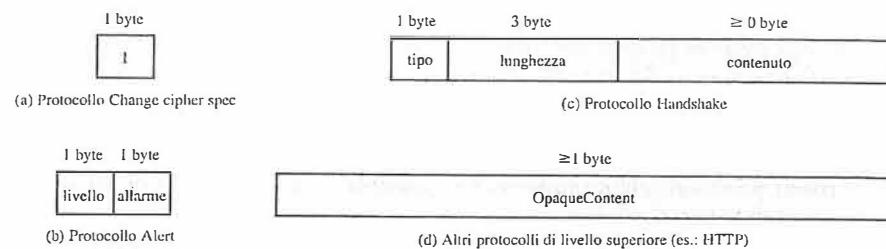


Figura 7.5 Payload del protocollo Record di SSL.

Protocollo Alert

Il protocollo Alert è utilizzato per comunicare all'unità prioritaria messaggi di allarme relativi a SSL. Come per le altre applicazioni che utilizzano SSL, i messaggi di allarme sono compressi e cifrati, come specificato nello stato corrente.

Ogni messaggio in questo protocollo è composto da due byte (Figura 7.5b). Il primo byte assume il valore warning (1) o fatal (2) per trasmettere la gravità del messaggio. Se il livello è fatal, SSL termina immediatamente la connessione. Le altre connessioni nella stessa sessione possono continuare, ma non si possono stabilire nuove connessioni nella sessione in questione. Il secondo byte contiene un codice che denota lo specifico allarme. Prima di tutto, vengono elencati gli allarmi che sono sempre fatali (le definizioni sono tratte dalla specifica SSL).

- **unexpected_message**: è stato ricevuto un messaggio non riconosciuto.
- **bad_record_mac**: è stato ricevuto un MAC errato.
- **decompression_failure**: la funzione di decompressione ha ricevuto un input improprio (per esempio, risulta incapace di effettuare la decompressione oppure effettua la decompressione a una lunghezza superiore a quella massima consentita).
- **handshake_failure**: il mittente non è stato capace di negoziare un insieme accettabile di parametri di sicurezza con le opzioni disponibili.
- **illegal_parameter**: un campo di un messaggio handshake conteneva un valore al di fuori dell'insieme dei valori ammissibili o era inconsistente con gli altri campi.

Gli allarmi rimanenti sono:

- **close_notify**: notifica al ricevente che il mittente non invierà più messaggi nella connessione corrente. Si richiede a ciascuna parte di inviare un allarme close_notify prima di chiudere la parte di scrittura di una connessione.

- **no_certificate**: può essere inviato in risposta a una richiesta di certificato quando non è disponibile alcun certificato adatto.
- **bad_certificate**: il certificato ricevuto era corrotto (per esempio, conteneva una firma non verificabile).
- **unsupported_certificate**: il certificato ricevuto ha un tipo non supportato.
- **certificate_revoked**: il certificato è stato revocato dal suo firmatario.
- **certificate_expired**: il certificato è scaduto.
- **certificate_unknown**: è intervenuto qualche altro problema, non noto durante l'elaborazione del certificato, che lo ha reso inaccettabile.

Protocollo Handshake

Il protocollo Handshake è la parte più complessa di SSL. Questo protocollo consente al client e al server di autenticarsi a vicenda e di negoziare un algoritmo di cifratura e di MAC e le chiavi di cifratura da utilizzare per la protezione dei dati inviati in un record SSL.

Il protocollo Handshake è utilizzato prima di spedire qualunque dato applicativo. Consiste in una serie di messaggi scambiati fra client e server. Tutti questi messaggi hanno il formato mostrato nella Figura 7.5c. Ciascun messaggio ha tre campi.

- **Tipo, 1 byte (type)**: indica un tipo fra 10 possibili tipi di messaggio, riportati nella Tabella 7.2.
- **Lunghezza, 3 byte (length)**: lunghezza del messaggio in byte.
- **Contenuto, ≥ 0 byte (content)**: parametri associati con il messaggio in questione, elencati nella Tabella 7.2.

La Figura 7.6 mostra lo scambio iniziale necessario a instaurare una connessione logica fra client e server. Lo scambio può essere suddiviso in quattro fasi.

Tipo di messaggio	Parametri
hello_request	Nessun parametro
client_hello	Versione, random, ID di sessione, suite di cifratura, metodo di compressione
server_hello	Versione, random, ID di sessione, suite di cifratura, metodo di compressione
certificate	Catena di certificati X.509v3
server_key_exchange	Parametri, firma
certificate_request	Tipo, autorità
server_done	Nessun parametro
certificate_verify	Firma
client_key_exchange	Parametri, firma
finished	Valore hash

Tabella 7.2 Tipi di messaggio del protocollo Handshake di SSL.

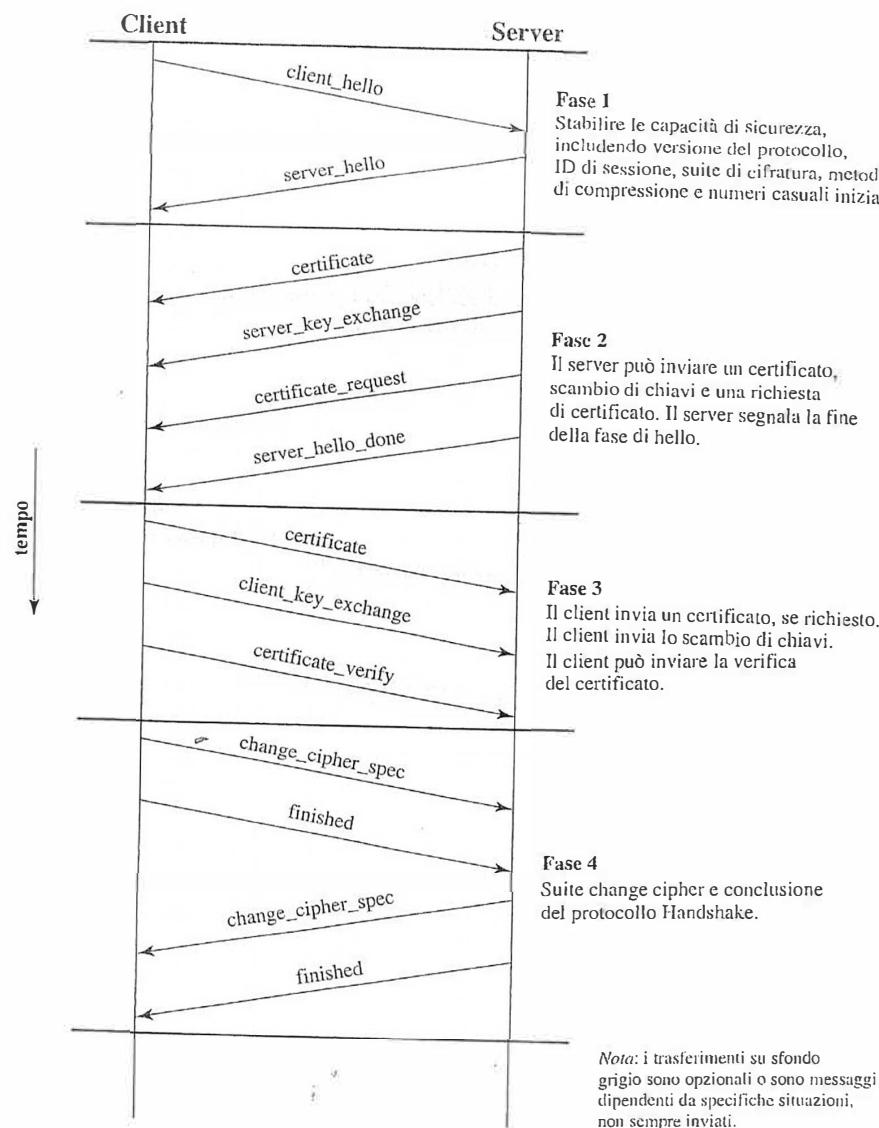


Figura 7.6 Funzionamento del protocollo Handshake.

Fase 1. Stabilire le capacità di sicurezza

Questa fase è usata per instaurare una connessione logica e per stabilire le capacità di sicurezza che saranno ad essa associate. Il client inizia lo scambio inviando un **messaggio client_hello** con i seguenti parametri.

- **Versione (version):** la versione più elevata di SSL supportata dal client.
- **Random:** struttura casuale generata dal client, che consiste in un timestamp a 32 bit e in 28 byte prodotti da un generatore di numeri casuali sicuro. Tali valori servono come nonce e sono utilizzati durante lo scambio delle chiavi per la prevenzione di attacchi di replay.
- **ID di sessione (session ID):** identificatore di sessione a lunghezza variabile. Un valore diverso da zero denota che il client vuole aggiornare i parametri di una connessione esistente o creare una nuova connessione nella sessione attuale. Un valore zero indica che il client vuole instaurare una nuova connessione in una nuova sessione.
- **Suite di cifratura (CipherSuite):** lista contenente le combinazioni di algoritmi crittografici supportati dal client, in ordine decrescente di preferenza. Ciascun elemento della lista (ovvero ciascuna suite di cifratura) definisce sia l'algoritmo per lo scambio delle chiavi sia una CipherSpec (entrambi descritti in seguito).
- **Metodo di compressione (compression method):** lista dei metodi di compressione supportati dal client.

Dopo aver inviato il messaggio **client_hello**, il client attende il **messaggio server_hello**, contenente gli stessi parametri del messaggio **client_hello**. Al messaggio **server_hello** si applicano le seguenti convenzioni. Il campo Versione contiene il valore di versione più basso suggerito dal client e il più elevato supportato dal server. Il campo Random viene generato dal server ed è indipendente dal campo Random del client. Se il campo ID di sessione del client era diverso da zero, lo stesso valore è usato dal server; altrimenti il campo ID di sessione del server contiene il valore di una nuova sessione. Il campo Suite di cifratura contiene una specifica suite di cifratura scelta dal server fra quelle proposte dal client. Il campo Compressione contiene il metodo di compressione scelto dal server fra quelli proposti dal client.

Il primo elemento del parametro Suite di cifratura è il metodo di scambio di chiavi (per esempio, lo strumento attraverso cui vengono scambiate le chiavi crittografiche per la cifratura convenzionale e il MAC). I metodi di scambio delle chiavi supportati sono i seguenti.

- **RSA.** La chiave segreta è cifrata con la chiave pubblica RSA del ricevente. Deve essere reso disponibile un certificato a chiave pubblica per la chiave del ricevente.
- **Fixed Diffie-Hellman.** Algoritmo di scambio di chiavi Diffie-Hellman in cui il certificato del server contiene i parametri pubblici Diffie-Hellman firmati dall'autorità certificativa (CA), cioè il certificato a chiave pubblica contiene i parametri della chiave pubblica di Diffie-Hellman. Il client fornisce i propri parametri di chiave pubblica Diffie-Hellman all'interno di un certificato se è richiesta l'autenticazione del client, oppure in un messaggio di scambio di chiavi. Questo metodo porta ad avere una chiave segreta stabilita fra due parti, basata sul calcolo Diffie-Hellman utilizzando le chiavi pubbliche fissate.

- **Ephemeral Diffie-Hellman.** Tecnica utilizzata per generare chiavi segrete temporanee (*one-time*). In questo caso, vengono scambiate le chiavi pubbliche Diffie-Hellman firmate con la chiave privata RSA o DSS del mittente. Il ricevente può utilizzare la chiave pubblica corrispondente per verificare la firma. I certificati sono utilizzati per autenticare le chiavi pubbliche. Questa sembrerebbe la più sicura delle tre opzioni Diffie-Hellman in quanto genera una chiave temporanea e autenticata.
- **Anonymous Diffie-Hellman.** Utilizza l'algoritmo Diffie-Hellman di base, senza autenticazione. Ovvero, ciascuna parte invia i propri parametri pubblici all'altra parte, senza autenticazione. Questo approccio è vulnerabile rispetto ad attacchi di tipo man-in-the-middle, in cui l'avversario esegue l'algoritmo anonymous Diffie-Hellman con entrambe le parti.
- **Fortezza.** Tecnica definita per lo schema Fortezza.

La definizione del metodo per lo scambio delle chiavi è la CipherSpec che comprende i seguenti campi.

- **CipherAlgorithm** (algoritmo di cifratura): qualunque algoritmo precedentemente menzionato (RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza).
- **MACAlgorithm** (algoritmo MAC): MD5 o SHA-1.
- **CipherType** (tipo di cifratura): a flusso o a blocchi.
- **IsExportable** (è esportabile): vero o falso.
- **HashSize** (dimensione hash): 0, 16 (per MD5) o 20 (per SHA-1) byte.
- **Key Material** (dati per le chiavi): sequenza di byte contenente i dati usati nella generazione delle chiavi di scrittura.
- **IV Size** (dimensione di IV): dimensione del vettore di inizializzazione per la cifratura in modalità CBC.

Fase 2. Autenticazione del server e scambio delle chiavi

Il server inizia questa fase inviando il proprio certificato, nel caso debba essere autenticato; il messaggio contiene un certificato X.509 o una catena di certificati. Il **messaggio certificate** è richiesto per qualunque metodo di scambio delle chiavi concordato, ad eccezione del metodo anonymous Diffie-Hellman. Si noti che se si usa fixed Diffie-Hellman, il messaggio certificate ha il ruolo di messaggio server_key_exchange in quanto comprende i parametri pubblici Diffie-Hellman del server.

Successivamente, si può inviare, se richiesto, un **messaggio server_key_exchange**. Non è richiesto in due casi: quando il server ha inviato un certificato con i parametri fixed Diffie-Hellman oppure quando si utilizza lo scambio di chiavi RSA. Il messaggio server_key_exchange è necessario nei seguenti casi.

- **Anonymous Diffie-Hellman.** Il contenuto del messaggio è composto dai due valori globali Diffie-Hellman (un numero primo e una radice primitiva del numero) e dalla chiave pubblica Diffie-Hellman del server (Figura 3.10).

- **Ephemeral Diffie-Hellman.** Il contenuto del messaggio comprende i tre parametri Diffie-Hellman forniti per anonymous Diffie-Hellman e la firma di questi parametri.
- **Scambio di chiavi RSA, in cui il server usa RSA ma ha una chiave RSA di sola firma.** Conformemente, il client non può semplicemente inviare una chiave segreta cifrata con la chiave pubblica del server. Al contrario, il server deve creare una copia temporanea chiave pubblica/chiave privata RSA e utilizzare il messaggio server_key_exchange per inviare la chiave pubblica. Il messaggio comprende i due parametri della chiave pubblica RSA temporanea (esponente e modulo, Figura 3.8) e una firma di questi parametri.

• Fortezza.

Come di consueto, una firma è creata considerando il codice hash di un messaggio e cifrandolo con la chiave privata del mittente. In questo caso, il codice hash è definito come:

```
hash(ClientHello.random || ServerHello.random || ServerParams)
```

Quindi il codice hash copre non solo i parametri Diffie-Hellman o RSA, ma anche i due nonce dei due messaggi hello iniziali, fornendo quindi garanzie contro attacchi di replay e di errata identificazione degli utenti. In caso di una firma DSS, il codice hash è calcolato mediante l'algoritmo SHA-1. Nel caso di una firma RSA, vengono calcolati i valori hash sia MD5 che SHA-1, e la loro concatenazione (36 byte) viene cifrata con la chiave privata del server.

Successivamente, un server che non usa anonymous Diffie-Hellman può richiedere un certificato al client. Il **messaggio certificate_request** comprende due parametri: certificate_type (tipo di certificato) e certificateAuthorities (autorità del certificato). Il tipo di certificato indica l'algoritmo a chiave pubblica e il suo impiego:

- RSA, solo firma
- DSS, solo firma
- RSA per fixed Diffie-Hellman (in questo caso la firma è usata solo per scopi di autenticazione, inviando un certificato firmato con RSA)
- DSS per fixed Diffie-Hellman (ugualmente usato solo per scopi di autenticazione)
- RSA per ephemeral Diffie-Hellman
- DSS per ephemeral Diffie-Hellman
- Fortezza

Il secondo parametro nel messaggio certificate_request è una lista di nomi di autorità certificate accettabili. L'ultimo messaggio della Fase 2, sempre richiesto, è il **messaggio server_done**, inviato dal server per segnalare il completamento del server hello e dei messaggi associati.

Dopo aver inviato questo messaggio, il server attende una risposta dal client. Questo messaggio non ha parametri.

Fase 3. Autenticazione del client e scambio delle chiavi

Alla ricezione del messaggio server_done, il client dovrebbe verificare che il server abbia fornito un certificato valido, se richiesto, e controllare che i parametri del messaggio server_hello siano accettabili. Se tutte queste condizioni sono soddisfatte, il client invia uno o più messaggi al server.

Se il server ha richiesto un certificato, il client inizia questa fase inviando un **messaggio certificate**. Il client invia invece un messaggio di allarme no_certificate qualora non fosse disponibile alcun certificato adatto.

Successivamente, deve essere inviato il messaggio **client_key_exchange**. Il contenuto del messaggio dipende dalla tipologia di scambio di chiavi, come segue.

- **RSA.** Il client genera un **valore segreto pre-master** di 48 byte e lo cifra con la chiave pubblica contenuta nel certificato del server o con la chiave RSA temporanea contenuta nel messaggio server_key_exchange. Il suo utilizzo nel calcolo del **valore master secreto** verrà spiegato in seguito.
- **Ephemeral o anonymous Diffie-Hellman.** Vengono inviati i parametri pubblici Diffie-Hellman del client.
- **Fixed Diffie-Hellman.** I parametri pubblici Diffie-Hellman del client sono stati inviati in un messaggio certificate; quindi, il contenuto di questo messaggio è nullo.
- **Fortezza.** Sono inviati i parametri Fortezza del client.

Infine, in questa fase, il client può inviare un **messaggio certificate_verify** per fornire verifica esplicita di un certificato del client. Questo messaggio è inviato solamente in seguito a un qualunque certificato del client avente capacità di firma (per esempio, tutti i certificati ad eccezione di quelli contenenti parametri fixed Diffie-Hellman). Questo messaggio firma un codice hash sulla base dei precedenti messaggi, come segue:

```
CertificateVerify.signature.md5_hash
MD5(master_secret || pad_2 || MD5(handshake_messages ||
    master_secret || pad_1));
Certificate.signature.sha_hash
SHA(master_secret || pad_2 || SHA(handshake_messages ||
    master_secret || pad_1));
```

dove: pad_1 e pad_2 corrispondono ai valori definiti in precedenza per il MAC; handshake_messages fa riferimento a tutti i messaggi del protocollo Handshake inviati o ricevuti a partire dal messaggio client_hello escluso il messaggio corrente; master_secret è il valore segreto calcolato in base a modalità spiegate successivamente in questo paragrafo. Se la chiave privata dell'utente è DSS, allora la si utilizza per la cifratura del valore hash SHA-1. Se la chiave privata dell'utente è RSA, allora la si utilizza per la cifratura della concatenazione dei valori hash MD5 e SHA-1. In entrambi i casi, lo scopo è quello di verificare, per il certificato del client, il possesso della chiave privata da parte del client. L'idea è che se anche qualcuno stesse facendo un cattivo uso del certificato del client, non sarebbe comunque in grado di inviare questo messaggio.

Fase 4. Conclusione

Questa fase termina l'instaurazione di una connessione sicura. Il client invia un **messaggio change_cipher_spec** e copia il valore CipherSpec dello stato in attesa nel valore CipherSpec corrente. Si noti che questo messaggio non è considerato parte del protocollo Handshake, ma viene inviato utilizzando il protocollo Change cipher spec. A questo punto il client invia immediatamente il **messaggio finished**, utilizzando i nuovi algoritmi, le chiavi e i valori segreti. Il messaggio finished verifica che i processi di scambio delle chiavi e di autenticazione abbiano avuto successo. Il contenuto del messaggio finished è la concatenazione di due valori hash:

```
MD5(master_secret || pad2 || MD5(handshake_messages ||
    Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
    Sender || master_secret || pad1))
```

dove Sender è un codice che identifica il fatto che il mittente è il client e handshake_messages corrisponde ai dati di tutti i messaggi di handshake fino a questo messaggio non compreso.

In risposta a questi due messaggi, il server invia il proprio messaggio change_cipher_spec, trasferisce il valore CipherSpec dello stato in attesa in quello corrente e invia il proprio messaggio finished. A questo punto, l'handshake è completo e client e server possono iniziare a scambiarsi dati a livello di applicazione.

Elaborazioni crittografiche

Altri due argomenti di particolare interesse sono la creazione di un valore master segreto condiviso mediante lo scambio di chiavi e la generazione dei parametri di crittografia a partire dal valore master segreto.

Creazione del valore master segreto

Il valore master segreto condiviso è un valore one-time di 48 byte (384 bit) generato per la sessione corrente mediante scambio sicuro di chiavi. La creazione avviene in due fasi. Prima di tutto viene scambiato un valore pre_master_secret; poi entrambe le parti calcolano il valore master_secret. Per lo scambio del valore pre_master_secret esistono due possibilità.

- **RSA.** Il client genera un valore pre_master_secret di 48 byte, cifrato con la chiave pubblica RSA del server, e lo invia al server. Il server decifra il testo cifrato ricevuto utilizzando la propria chiave privata per ricostruire il valore pre_master_secret.
- **Diffie-Hellman.** Sia il client che il server generano una chiave pubblica Diffie-Hellman. Dopo aver scambiato tali chiavi, ciascuna parte esegue il calcolo Diffie-Hellman per creare il valore condiviso pre_master_secret.

Entrambe le parti calcolano il valore master_secret come segue:

```
master_secret = MD5(pre_master_secret || SHA('A' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
MD5(pre_master_secret || SHA('BB' ||
    pre_master_secret || ClientHello.random ||
```

(segue da p. prec.)

```

        ServerHello.random) ||

        MD5(pre_master_secret || SIIA('CCC' ||
            pre_master_secret || ClientHello.random ||
            ServerHello.random))
    
```

dove ClientHello.random e ServerHello.random sono i due nonce scambiati nei messaggi hello iniziali.

Generazione dei parametri di crittografia

Le CipherSpec richiedono un valore segreto MAC di scrittura, una chiave di scrittura e un vettore IV di scrittura sia per il client che per il server, generati in tale ordine a partire dal valore master segreto. Questi parametri sono generati dal valore master segreto trasformandolo mediante un algoritmo hash in una sequenza di byte di lunghezza sufficiente a contenere tutti i parametri.

La generazione delle chiavi a partire dal valore master segreto utilizza lo stesso formato usato nella generazione del valore master segreto a partire dal valore segreto premaster:

```

key_block = MD5(master_secret || SHA('A' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('BB' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('CCC' || master_
    secret || ServerHello.random ||
    ClientHello.random)) ...
    
```

finché viene generato un risultato di lunghezza sufficiente. Il risultato di questa struttura algoritmica è una funzione pseudocasuale. Si può considerare il valore master_secret come valore seme pseudocasuale per la funzione. I numeri casuali del client e del server possono essere considerati come valori salt per rendere più difficile la crittoanalisi (cfr Capitolo 9 per l'uso di valori salt).

TLS

TLS è un'iniziativa di standardizzazione IETF il cui obiettivo è quello di produrre una versione Internet standard di SSL. La versione corrente di Proposed standard di TLS, definita nel documento RFC 2246, è molto simile a SSLv3. In questo paragrafo vengono evidenziate le differenze.

Numero di versione

Il formato record di TLS è lo stesso di SSL (Figura 7.4) e i campi nell'intestazione hanno lo stesso significato. La sola differenza risiede nei valori del campo versione. Nella versione corrente di TLS, la versione principale corrisponde a 3 e la versione minore a 1.

MAC

Ci sono due differenze fra gli schemi MAC di SSLv3 e TLS: l'algoritmo utilizzato e l'ambito dell'elaborazione MAC. TLS utilizza l'algoritmo HMAC definito nel documento RFC 2104. Dal Capitolo 3, ricordiamo che HMAC è definito come segue:

$$\text{HMAC}_K(M) = \text{H}[(K^* \oplus \text{opad}) \parallel \text{H}[(K^* \oplus \text{ipad}) \parallel M]]$$

dove:

- H = funzione hash contenuta (per TLS, MD5 o SHA-1)
- M = messaggio in ingresso a HMAC
- K^* = chiave segreta completata con una serie di zero aggiuntivi a sinistra per produrre un risultato uguale alla lunghezza del blocco utilizzato nella codifica hash (per MD5 o SHA-1, la lunghezza del blocco è 512 bit)
- ipad = 00110110 (36 in esadecimale) ripetuto 64 volte (512 bit)
- opad = 01011100 (5C in esadecimale) ripetuto 64 volte (512 bit)

SSLv3 usa lo stesso algoritmo ad eccezione del fatto che i byte di completamento sono concatenati con la chiave segreta piuttosto che essere combinati mediante OR esclusivo con la chiave segreta, a sua volta completata per raggiungere la necessaria lunghezza del blocco. Il livello di sicurezza dovrebbe essere lo stesso in entrambi i casi.

Per TLS, il calcolo del MAC comprende i campi riportati nella seguente espressione:

```

HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
    TLSCompressed.version || TLSCompressed.length ||
    TLSCompressed.fragment)
    
```

Il calcolo del MAC comprende tutti i campi coperti dal corrispondente calcolo di SSLv3, con in più il campo TLSCompressed.version, che indica la versione del protocollo in uso.

Funzione pseudocasuale

TLS fa uso di una funzione pseudocasuale conosciuta come **PRF** (*pseudorandom function*) per espandere i valori segreti in blocchi di dati per la generazione o validazione delle chiavi. L'obiettivo è utilizzare un valore segreto condiviso relativamente piccolo ma generare blocchi di dati più lunghi, in modo che l'approccio sia sicuro rispetto alle tipologie di attacchi perpetrati a funzioni hash e MAC. La funzione PRF è basata sulla seguente funzione di espansione dei dati (Figura 7.7):

```

P_hash(valore_segreto, seme) = HMAC_hash(valore_segreto, A(1) || seme) ||
    HMAC_hash(valore_segreto, A(2) || seme) ||
    HMAC_hash(valore_segreto, A(3) || seme) || ...
    
```

Dove A() è definita come:

- A(0) = seme
- A(i) = HMAC_hash (valore_segreto, A($i - 1$))

La funzione di espansione dei dati usa l'algoritmo HMAC, con MD5 o SHA-1 come funzioni hash sottostanti. Come si può notare, P_hash può essere iterata tante volte quante sono quelle necessarie a produrre la quantità di dati richiesta. Ad esempio, se si fosse utilizzata P_SHA-1 per generare 64 byte di dati, sarebbe stata iterata quattro volte, producendo 80 byte di dati, di cui gli ultimi 16 non sarebbero stati considerati. In questo caso, anche P_MD5 sarebbe stata iterata quattro volte, producendo esattamente 64 byte di dati. Si noti che ogni iterazione richiede due esecuzioni di HMAC, ciascuna delle quali richiede a sua volta due esecuzioni degli algoritmi hash sottostanti.

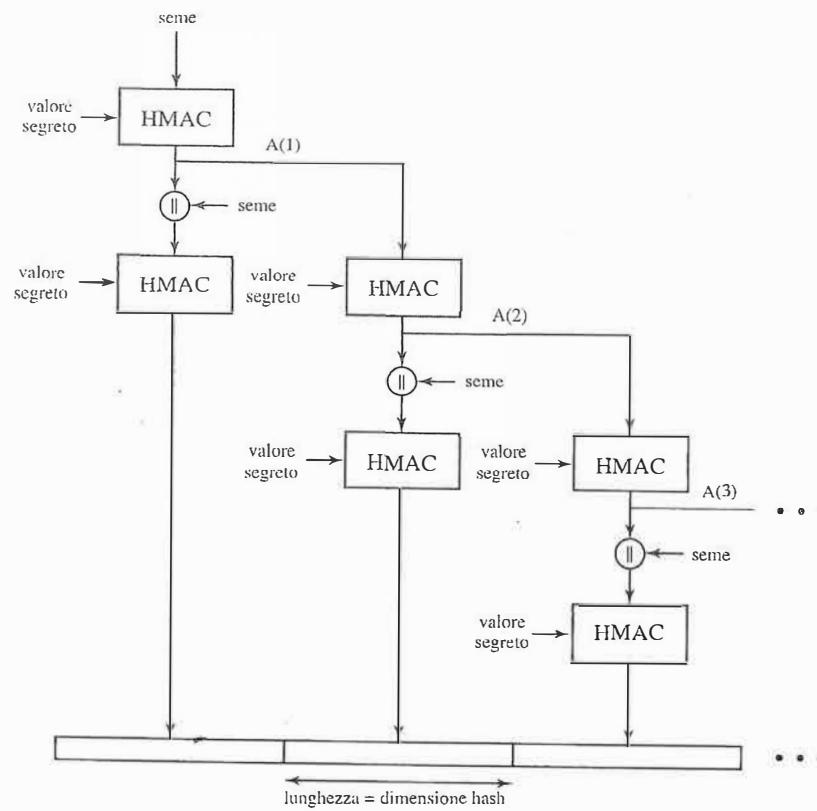


Figura 7.7 Funzione P_hash (valore segreto, seme) di TLS.

Per aumentare il livello di sicurezza della funzione PRF, si usano due algoritmi hash in modo che se uno dei due algoritmi rimane sicuro, la funzione dovrebbe essere sicura. PRF è definita come:

```
PRF(valore segreto, etichetta, seme) = P_MD5(S1, etichetta || seme) ⊕  
P_SHA-1(S2, etichetta || seme)
```

PRF riceve in ingresso un valore segreto, un'etichetta identificativa e un valore di seme e produce un risultato di lunghezza arbitraria. Il risultato è creato dividendo il valore segreto in parti uguali (S_1 e S_2) ed eseguendo la funzione P_hash su ciascuna di esse, utilizzando MD5 su una metà e SHA-1 sull'altra. I due risultati sono combinati mediante OR esclusivo per produrre il risultato finale; a tale scopo, P_MD5 deve essere iterata generalmente più volte di P_SHA-1 per produrre una pari quantità di dati in ingresso alla funzione di OR esclusivo.

Codici di allarme

TLS supporta tutti i codici di allarme definiti in SSLv3 ad eccezione del codice no_certificate. In TLS sono definiti un certo numero di codici aggiuntivi; fra questi, i seguenti denotano situazioni sempre fatali.

- **decryption_failed**: decifratura non valida di un testo cifrato; o non era un multiplo pari della lunghezza del blocco oppure i valori di completamento, in fase di verifica, risultavano errati.
- **record_overflow**: è stato ricevuto un record TLS con un payload (testo cifrato) avente lunghezza superiore a $2^{14} + 2048$ byte, oppure il testo cifrato è stato decifrato a una lunghezza superiore a $2^{14} + 1024$ byte.
- **unknown_ca**: è stata ricevuta una catena di certificati valida o una catena parziale, ma il certificato non è stato accettato in quanto il certificato della CA non può essere localizzato o non può essere confrontato con una CA conosciuta e fidata.
- **access_denied**: è stato ricevuto un certificato valido, ma nell'applicare il controllo dell'accesso, il mittente ha deciso di non procedere con la negoziazione.
- **decode_error**: non è possibile decodificare un messaggio in quanto un campo contiene un valore al di fuori dell'intervallo di valori ammissibili specificato, oppure la lunghezza del messaggio era errata.
- **export_restriction**: è stata scoperta una negoziazione non ottemperante alle restrizioni di esportazione circa la lunghezza della chiave.
- **protocol_version**: la versione di protocollo che il client ha tentato di negoziare è riconosciuta ma non supportata.
- **insufficient_security**: valore ritornato al posto di handshake_failure quando una negoziazione è fallita per il fatto che il server richiede cifrari più sicuri di quelli supportati dal client.
- **internal_error**: un errore interno non connesso alla parte o alla correttezza del protocollo rende impossibile la continuazione.

I nuovi allarmi rimanenti sono i seguenti.

- **decrypt_error**: un'operazione di handshake di cifratura è fallita, comprendendo i casi di incapacità di verificare una firma, di decifrare uno scambio di chiavi o di validare un messaggio finished.
- **user_canceled**: l'operazione di handshake corrente viene cancellata per ragioni non connesse a un malfunzionamento del protocollo.
- **no_renegotiation**: inviato da un client in risposta a una richiesta hello oppure dal server in risposta a un messaggio hello del client dopo l'handshake iniziale. Un messaggio qualunque di questi porta normalmente alla rinegoziazione, ma questo allarme indica che il mittente non è in grado di effettuarla. Questo messaggio è sempre un messaggio di avviso.

Suite di cifratura

Esistono numerose, piccole differenze fra le suite di cifratura disponibili in SSLv3 e TLS.

- **Scambio di chiavi:** TLS supporta tutte le tecniche di scambio di chiavi di SSLv3 ad eccezione di Fortezza.
- **Algoritmi di cifratura simmetrica:** TLS comprende tutti gli algoritmi di cifratura simmetrica di SSLv3, ad eccezione di Fortezza.

Tipologie di certificato del client

TLS definisce le seguenti tipologie di certificati da richiedere in un messaggio certificate_request: rsa_sign, dss_sign, rsa_fixed_dh, e dss_fixed_dh. Tutti questi sono definiti in SSLv3. Inoltre, SSLv3 comprende rsa_ephemeral_dh, dss_ephemeral_dh, e fortezza_keas. Ephemeral Diffie-Hellman richiede la firma dei parametri Diffie-Hellman con RSA o DSS; per TLS, i tipi usati sono rsa_sign e dss_sign. Non è necessario un tipo separato di firma per firmare i parametri Diffie-Hellman. TLS non comprende lo schema Fortezza.

Messaggi certificate_verify e finished

Nel messaggio certificate_verify di TLS, i codici hash MD5 e SHA-1 sono calcolati solamente su handshake_messages. Nel caso di SSLv3, il calcolo del valore hash comprende anche il valore master_secret e i pad. L'impressione era che questi campi addizionali non aggiungessero ulteriore sicurezza.

Come per il messaggio finished in SSLv3, anche in TLS questo messaggio è un codice hash basato sul valore condiviso master_secret, sui messaggi handshake precedenti e su un'etichetta che identifica il client o il server. Il calcolo differisce leggermente. Nel caso di TLS, si ha:

```
PRF(master_secret, finished_label, MD5(handshake_messages) ||
     SHA-1(handshake_messages))
```

dove finished_label è la stringa "client finished" per il client e "server finished" per il server.

Elaborazioni di crittografia

Il valore pre_master_secret di TLS viene calcolato allo stesso modo del corrispondente valore di SSLv3. Come in SSLv3, il valore master_secret di TLS è calcolato come una funzione hash del valore pre_master_secret e dei due numeri casuali di hello. L'elaborazione in TLS differisce da quella di SSLv3 ed è definita come segue:

```
master_secret = PRF(pre_master_secret, "master secret",
                      clientHello.random || serverHello.random)
```

L'algoritmo viene eseguito fino a produrre un risultato pseudocasuale di 48 byte. Il calcolo del blocco di informazioni relative alle chiavi (chiavi segrete MAC, chiavi di cifratura di sessione e IV) è definito come segue:

```
key_block = PRF(master_secret, "key expansion",
                 SecurityParameters.server_random ||
                 SecurityParameters.client_random)
```

finché non viene generato un risultato di dimensione sufficiente. Come per SSLv3, key_block è una funzione del valore master_secret e dei numeri casuali del client e del server, ma l'algoritmo di TLS è diverso.

Completamento

In SSL, il completamento aggiunto prima di effettuare la cifratura dei dati utente corrisponde alla quantità minima necessaria a rendere la dimensione totale dei dati da cifrare multiplo della lunghezza del blocco del cifrario. In TLS, il completamento può essere una quantità qualunque che porti ad avere un risultato multiplo della lunghezza del blocco del cifrario, fino a un massimo di 255 byte. Ad esempio, se il testo in chiaro (o quello compreso se si usa la compressione), insieme con il MAC e il byte padding_length hanno una lunghezza totale di 79 byte, allora la lunghezza in byte del completamento può essere 1, 9, 17 e così via, fino a 249. Una lunghezza variabile per il completamento può essere utilizzata per scorggiare attacchi basati sull'analisi della lunghezza dei messaggi scambiati.

7.3 Secure electronic transaction (SET)

SET è una specifica aperta di cifratura e sicurezza progettata per fornire protezione alle transazioni effettuate con carta di credito su Internet. La versione corrente, SETv1, è il frutto di una richiesta di standard di sicurezza avanzata da MasterCard e Visa nel febbraio 1996. Nello sviluppo della specifica iniziale erano coinvolte numerose aziende fra cui IBM, Microsoft, Netscape, RSA, Terisa e VeriSign. A partire dal 1996, sono state effettuate numerose verifiche della specifica iniziale e nel 1998 era disponibile la prima serie di prodotti rispondenti alle specifiche SET.

SET non è di per sé un sistema di pagamento; consiste piuttosto in un insieme di protocolli e formati di sicurezza che consentono agli utenti di utilizzare, in maniera sicura, l'infrastruttura di pagamento basata su carta di credito valida su una rete aperta, come ad esempio Internet. Essenzialmente, SET fornisce tre servizi:

- un canale di comunicazione sicuro fra le parti coinvolte in una transazione;
- affidabilità attraverso l'uso di certificati digitali X.509v3;
- riservatezza, in quanto l'informazione è disponibile solo per le parti coinvolte in una transazione, quando e dove necessario.

SET è una specifica complessa definita in tre volumi pubblicati nel maggio 1997:

- **Volume 1:** Business Description (80 pagine)
- **Volume 2:** Programmer's Guide (629 pagine)
- **Volume 3:** Formal Protocol Definition (262 pagine)

per un totale di 971 pagine di specifica. Per contro, la specifica SSLv3 è composta da 63 pagine e quella TLS ne conta 71. Di conseguenza, in questo paragrafo si riporta una sintesi di questa articolata specifica.

Descrizione generale di SET

Iniziamo la trattazione di SET considerando, nell'ordine, i requisiti dal punto di vista commerciale, le caratteristiche principali e i partecipanti alle transazioni SET.

Requisiti

Il Volume 1 della specifica SET elenca i seguenti requisiti commerciali per l'elaborazione sicura dei pagamenti con carte di credito su Internet e altre reti.

- **Fornire riservatezza per il pagamento e per le informazioni sugli ordini.** È necessario garantire ai proprietari di carta di credito che queste informazioni siano sicure e accessibili solo da parte dei riceventi voluti. La riservatezza riduce anche il rischio di frode perpetrata da una delle due parti partecipanti alla transazione o da terze parti con intenti dolosi. Per fornire riservatezza, SET utilizza tecniche di cifratura.
- **Garantire l'integrità di tutti i dati trasmessi.** Significa garantire che non si verifichino cambiamenti nel contenuto dei messaggi SET durante la loro trasmissione. L'integrità è fornita mediante meccanismi di firma digitale.
- **Fornire autenticazione riguardo al fatto che il proprietario della carta di credito è l'utente legittimo del conto associato a quella carta.** Il meccanismo che collega il proprietario della carta a uno specifico numero di conto riduce l'incidenza delle frodi e il costo complessivo dell'elaborazione dei pagamenti. Per verificare che il proprietario della carta di credito sia un utente legittimo di un conto valido, si utilizzano firme digitali e certificati.
- **Fornire autenticazione riguardo al fatto che un commerciante possa accettare transazioni con carta di credito attraverso la sua relazione con l'Istituto di credito.** Completa il requisito precedente. I proprietari di carte di credito devono poter identificare i commercianti con cui possono condurre transazioni sicure. Anche a tale scopo, si utilizzano firme digitali e certificati.
- **Garantire che vengano utilizzate le migliori soluzioni di sicurezza e le migliori tecniche di progettazione di sistema al fine di proteggere tutte le parti legittime coinvolte in una transazione di commercio elettronico.** SET è una specifica ben testata, basata su algoritmi di crittografia e protocolli ad elevata sicurezza.
- **Creare un protocollo che non dipenda dai meccanismi di sicurezza a livello di trasporto e che non ne impedisca l'uso.** SET può operare in maniera sicura al di sopra di uno stack TCP/IP "puro". Tuttavia, SET non interferisce con l'uso di altri meccanismi di sicurezza quali IPSec e SSL/TLS.
- **Facilitare e incoraggiare l'interoperabilità fra fornitori di software e di reti.** I protocolli e i formati SET sono indipendenti dalla piattaforma hardware, dal sistema operativo e dal software web.

Caratteristiche rilevanti di SET

Per soddisfare i requisiti appena presentati, SET incorpora le seguenti caratteristiche.

- **Riservatezza delle informazioni.** Le informazioni relative al conto del titolare della carta e al pagamento sono protette durante il trasferimento sulla rete. Una caratteristica interessante e importante di SET è che impedisce al commerciante di venire a conoscenza del numero di carta di credito del titolare; tale informazione viene fornita solamente alla banca di emissione. La riservatezza viene realizzata mediante cifratura convenzionale con DES.
- **Integrità dei dati.** Le informazioni relative al pagamento inviate dai titolari di carta di credito ai commercianti comprendono specifiche sull'ordine, dati personali e istruzioni di pagamento. SET garantisce che i messaggi contenenti queste informazioni non siano modificati durante la trasmissione. L'integrità dei messaggi è realizzata mediante l'utilizzo di firme digitali RSA, con codici hash SHA-1. Alcuni messaggi sono protetti anche mediante HMAC utilizzando SHA-1.
- **Autenticazione del conto del titolare di carta di credito.** SET permette ai commercianti di verificare che un titolare di carta di credito sia un utente legittimo di un conto valido per quella carta. A tale scopo SET utilizza certificati digitali X.509v3 con firme RSA.
- **Autenticazione del commerciante.** SET permette ai titolari di carta di verificare che il commerciante abbia un qualche rapporto con un istituto finanziario che gli consenta di accettare carte di pagamento. A tale scopo SET usa certificati digitali X.509v3 con firme RSA.

Si noti che, a differenza di IPSec e SSL/TLS, SET fornisce solo una scelta per ciascun algoritmo di cifratura. Ciò avviene in quanto SET è un'unica applicazione con un unico insieme di requisiti, mentre IPSec e SSL/TLS sono stati progettati per supportare molteplici applicazioni.

Partecipanti SET

La Figura 7.8 mostra i partecipanti nel sistema SET.

- **Titolare di carta di credito (cardholder).** In ambiente elettronico, consumatori e acquirenti interagiscono con i commercianti tramite PC su Internet. Un titolare è un possessore autorizzato di una carta di credito per effettuare pagamenti (per esempio: MasterCard, Visa) emessa da un istituto emittente.
- **Commerciante (merchant).** Un commerciante è una persona o un'organizzazione che possiede beni o servizi da vendere ai titolari di carta di credito. Generalmente, i beni e i servizi sono offerti attraverso un sito web oppure attraverso posta elettronica. Un commerciante che accetta carte di credito per i pagamenti deve avere un rapporto con un'istituto acquirente.

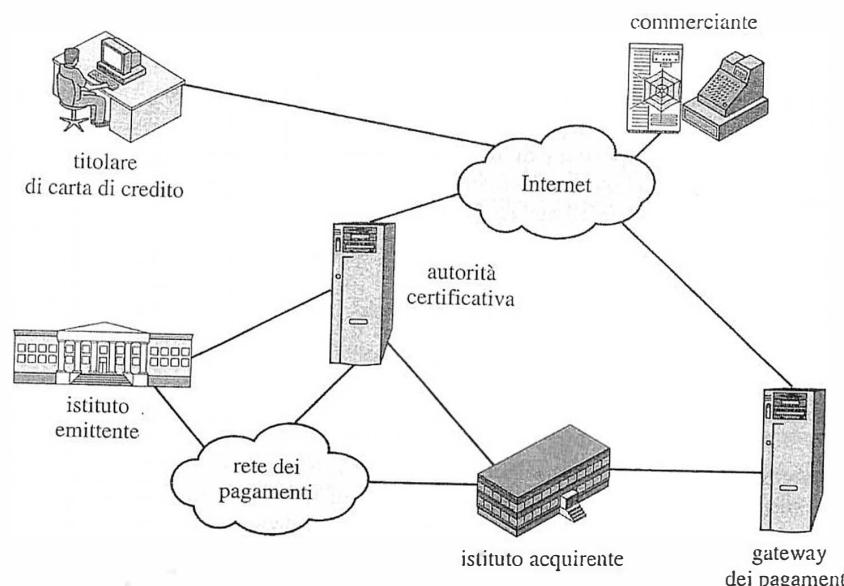


Figura 7.8 Partecipanti SET.

- Istituto emittente (issuer).** Istituto finanziario, come ad esempio una banca, che fornisce una carta di pagamento al titolare di carta. In definitiva, l'istituto emittente è il responsabile del pagamento del debito del titolare di carta.
- Istituto acquirente (acquirer).** Istituto finanziario che stabilisce un conto con il commerciante ed elabora le autorizzazioni delle carte di pagamento e i pagamenti stessi. I commercianti generalmente accettano carte di credito di diversi istituti emittenti ma non vogliono dover trattare con molteplici associazioni bancarie o con svariati istituti emittenti singolarmente. L'istituto acquirente fornisce l'autorizzazione a un commerciante per il pagamento, sulla base del fatto che un certo conto di carta di credito è attivo e che l'acquisto proposto non eccede il limite di credito previsto per tale conto. L'istituto acquirente fornisce anche il servizio di trasferimento elettronico dei pagamenti sul conto del commerciante. Successivamente, l'istituto acquirente viene rimborsato dall'istituto emittente attraverso qualche tipologia di rete di pagamento per il trasferimento elettronico di fondi.
- Gateway dei pagamenti.** Funzione che elabora i messaggi di pagamento del commerciante; tale funzione è svolta dall'istituto acquirente stesso o da una terza parte designata. Il gateway dei pagamenti agisce da interfaccia fra SET e le reti di pagamento bancarie esistenti per le funzioni di autorizzazione e pagamento. Il commerciante scambia su Internet messaggi SET con il gateway dei pagamenti, mentre il gateway dei pagamenti possiede una connessione diretta o una connessione di rete con il sistema di elaborazione finanziaria dell'istituto acquirente.

- Autorità certificativa (CA).** Entità affidabile per l'emissione di certificati a chiave pubblica X.509v3 per i titolari di carta, i commercianti e i gateway dei pagamenti. Il successo di SET dipenderà dall'esistenza di un'infrastruttura di CA disponibile per questi scopi. Come visto nei capitoli precedenti, si utilizza una gerarchia di CA, cosicché non è necessario che tutti i partecipanti siano certificati da un'autorità radice.

Descriviamo ora brevemente la sequenza di eventi richiesti per una transazione. Successivamente, saranno trattati alcuni dettagli di crittografia.

- Il cliente apre un conto.** Il cliente ottiene un conto di carta di credito come, ad esempio, MasterCard o Visa, presso una banca che supporti pagamenti elettronici e SET.
- Il cliente riceve un certificato.** Dopo opportuna verifica dell'identità, il cliente riceve un certificato digitale X.509v3, firmato dalla banca. Il certificato verifica la chiave pubblica RSA del cliente e la sua scadenza. Stabilisce anche un rapporto, garantito dalla banca, fra la coppia di chiavi del cliente e la sua carta di credito.
- I commercianti possiedono i propri certificati.** Un commerciante che accetta una certa marca di carta di credito deve essere in possesso di due certificati per le due chiavi pubbliche da lui possedute: una per la firma dei messaggi e una per lo scambio di chiavi. Il commerciante necessita anche di una copia del certificato a chiave pubblica del gateway dei pagamenti.
- Il cliente effettua un ordine.** Questo processo può coinvolgere dapprima il cliente che naviga nel sito web del commerciante per scegliere gli articoli e calcolare il prezzo. Il cliente invia poi al commerciante una lista di articoli da acquistare. Il commerciante restituisce un modulo d'ordine contenente la lista degli articoli, il loro prezzo, il prezzo totale e un numero d'ordine.
- Validazione del commerciante.** Oltre al modulo d'ordine, il commerciante invia una copia del proprio certificato, in modo che il cliente possa verificare la validità del negozio con cui sta trattando.
- Invio dell'ordine e del pagamento.** Congiuntamente al proprio certificato, il cliente invia al commerciante informazioni sia sull'ordine sia sul pagamento. L'ordine conferma l'acquisto degli articoli elencati nel modulo d'ordine. Il pagamento contiene i dettagli della carta di credito. L'informazione di pagamento è cifrata in modo che non possa essere letta dal commerciante. Il certificato spedito dal cliente consente al commerciante di verificare, a sua volta, il cliente.
- Il commerciante richiede l'autorizzazione sul pagamento.** Il commerciante invia l'informazione di pagamento al gateway dei pagamenti, richiedendo l'autorizzazione al pagamento sulla base del fatto che il credito del cliente sia sufficiente a coprire l'acquisto in corso.
- Il commerciante conferma l'ordine.** Il commerciante invia la conferma dell'ordine al cliente.
- Il commerciante fornisce i prodotti o il servizio.** Il commerciante spedisce i prodotti o fornisce il servizio al cliente.
- Il commerciante richiede il pagamento.** Questa richiesta è inviata al gateway dei pagamenti, che gestisce tutta l'elaborazione del pagamento.

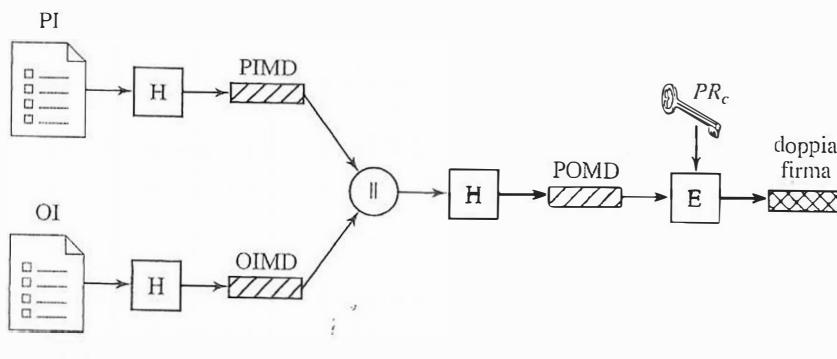
Doppia firma

Prima di entrare nel merito del protocollo SET, presentiamo un'importante innovazione introdotta in SET: la doppia firma. Lo scopo della doppia firma è quello di collegare due messaggi destinati a due diversi riceventi. In questo caso, il cliente vuole inviare informazioni sull'ordine (OI, *order information*) al commerciante e informazioni sul pagamento (PI, *payment information*) alla banca. Il commerciante non ha necessità di conoscere il numero di carta di credito del cliente e alla banca non occorre conoscere i dettagli dell'ordine effettuato dal cliente. Mantenendo queste due informazioni separate, il cliente dispone di una protezione aggiuntiva in termini di riservatezza. Tuttavia, le due informazioni devono essere collegate in modo da poter essere utilizzate in caso di controversie. Il collegamento è necessario per fare in modo che il cliente possa provare che il pagamento è destinato all'ordine in questione e non a qualche altro bene o servizio.

Per dimostrare le necessità del collegamento, si supponga che i clienti inviano al commerciante due messaggi: un messaggio OI firmato e un messaggio PI firmato. Si assuma, inoltre, che il commerciante inoltri il messaggio PI alla banca. Se il commerciante potesse intercettare un altro messaggio OI di questo stesso cliente, potrebbe sostenere che quest'ultimo OI – anziché quello originario – debba essere riferito al messaggio PI. Il collegamento impedisce il verificarsi di situazioni di questo genere.

La Figura 7.9 mostra l'impiego di una doppia firma per soddisfare i requisiti esposti nel paragrafo precedente. Il cliente ottiene il codice hash (utilizzando SHA-1) del messaggio PI e quello del messaggio OI. Questi due codici hash sono poi concatenati e si considera il codice hash del risultato. Infine, il cliente effettua la cifratura del valore hash finale mediante la propria chiave privata di firma, creando la doppia firma. L'operazione può essere schematizzata come segue:

$$DS = E(PR_c, [H(H(PI)||H(OI))])$$



PI = informazioni sul pagamento
OI = informazioni sull'ordine
H = funzione hash
|| = operatore di concatenazione

PIMD = digest del messaggio PI
OIMD = digest del messaggio OI
POMD = digest del messaggio di ordine pagamento
E = cifratura (RSA)
PR_c = chiave privata di firma del cliente

Figura 7.9 Generazione della doppia firma.

dove PR_c è la chiave privata di firma del cliente. Si supponga ora che il commerciante sia in possesso della doppia firma (DS, *dual signature*), del messaggio OI, e del digest di messaggio del messaggio PI (PIMD, *PI message digest*). Il commerciante possiede anche la chiave pubblica del cliente, ottenuta dal certificato del cliente. Di conseguenza il commerciante può calcolare i seguenti valori:

$$H(PIMD||H[OI]); D(PU_c, DS)$$

dove PU_c è la chiave pubblica di firma del cliente. Se questi due valori sono uguali, allora il commerciante ha verificato la firma. In maniera analoga, se la banca possiede DS, PI, il digest di messaggio per OI (OIMD, *OI message digest*) e la chiave pubblica del cliente, allora può calcolare:

$$H(H[OI]||OIMD); D(PU_c, DS)$$

Anche in questo caso, se i due valori sono uguali, la banca ha verificato la firma. In sintesi:

1. il commerciante ha ricevuto OI e ha verificato la firma;
2. la banca ha ricevuto PI e ha verificato la firma;
3. il cliente ha collegato OI e PI e può provare il collegamento.

Ad esempio, si supponga che il commerciante voglia sostituire, a proprio vantaggio, l'OI della transazione corrente con un altro OI. Dovrebbe trovare un altro OI il cui codice hash uguaglia l'OIMD esistente. Con SHA-1, ciò è ritenuto non fattibile. Pertanto, il commerciante non può collegare un diverso OI al PI della transazione corrente.

Elaborazione del pagamento

Nella Tabella 7.3, sono elencate le tipologie di transazioni supportate da SET. In seguito, si prenderanno in considerazione le seguenti transazioni:

- Richiesta di acquisto
- Autorizzazione di pagamento
- Riscossione del pagamento

Richiesta di acquisto

Prima che lo scambio relativo alla richiesta di acquisto cominci, il titolare della carta ha completato le attività di navigazione, selezione e ordine. Questa fase preliminare termina nel momento in cui il commerciante invia al cliente un modulo d'ordine compilato. Tutte queste attività si svolgono senza l'uso di SET.

Lo scambio relativo alla richiesta di acquisto è composto da quattro messaggi:

- Initiate Request (richiesta di inizio)
- Initiate Response (risposta di inizio)
- Purchase Request (richiesta di acquisto)
- Purchase Response (risposta di acquisto).

Registrazione del titolare di carta di credito	I titolari devono registrarsi presso una CA prima di poter inviare messaggi SET ai commercianti.
Registrazione del commerciante	I commercianti devono registrarsi presso una CA prima di poter scambiare messaggi SET con i clienti e i gateway dei pagamenti.
Richiesta di acquisto	Un messaggio inviato dal cliente al commerciante contenente l'OI destinato al commerciante e il PI destinato alla banca.
Autorizzazione di pagamento	Scambio tra il commerciante e il gateway dei pagamenti per autorizzare l'importo di un acquisto su un determinato conto di carta di credito.
Riscossione del pagamento	Consente al commerciante di richiedere un pagamento al gateway dei pagamenti.
Interrogazione sui certificati e sullo stato	Se la CA non è in grado di completare rapidamente l'elaborazione di una richiesta di certificato, invierà un messaggio di risposta al titolare di carta o al commerciante, specificando che il richiedente dovrà controllare la propria richiesta successivamente. Il titolare di carta o il commerciante invia il messaggio <i>Certificate Inquiry</i> per stabilire lo stato della propria richiesta di certificato e, se la richiesta è stata approvata, per ricevere il certificato stesso.
Interrogazione sugli acquisti	Consente al titolare di carta di verificare lo stato di avanzamento di un ordine dopo che il messaggio <i>Purchase Response</i> è stato ricevuto. Si noti che tale messaggio non fornisce informazioni sullo stato dei prodotti, ma indicazioni per quanto riguarda lo stato dell'autorizzazione e lo stato delle elaborazioni per la riscossione del pagamento e del credito.
Annullo dell'autorizzazione	Consente al commerciante di modificare precedenti richieste di autorizzazione. Se l'ordine non sarà completato, il commerciante annullerà l'intera autorizzazione. Se solo una parte dell'ordine non sarà completata (come quando alcune merci rimangono inesistenti), il commerciante annullerà l'autorizzazione solo per una parte dell'importo.
Annullo della riscossione	Consente al commerciante di correggere errori presenti nelle richieste di rilevazione quali, ad esempio, importi errati.
Credito	Consente al commerciante di trasferire un importo sul conto di un titolare di carta, nei casi in cui le merci acquistate vengano restituite o danneggiate durante la consegna. Il messaggio SET Credit è sempre inviato da un commerciante e non da un titolare di carta di credito. Tutte le comunicazioni fra titolare di carta e commerciante originate da un messaggio SET Credit vengono effettuate al di fuori di SET.
Annullo del credito	Consente al commerciante di correggere precedenti richieste di credito.
Richiesta di certificato del gateway dei pagamenti	Consente al commerciante di richiedere al gateway dei pagamenti una copia degli attuali certificati di scambio di chiavi e di firma.
Amministrazione dei lotti	Consente al commerciante di comunicare al gateway dei pagamenti informazioni riguardanti pagamenti di lotti.
Messaggio di errore	Indica che il ricevente rifiuta un messaggio inviato perché non verifica i controlli sul formato o sul contenuto.

Tabella 7.3 Tipi di transazione SET.

Al fine di inviare messaggi SET al commerciante, il titolare di carta deve avere una copia dei certificati del commerciante e del gateway dei pagamenti. Il cliente richiede i certificati nel messaggio **Initiate Request**, inviato al commerciante. Questo messaggio comprende la marca della carta di credito che il cliente sta utilizzando. Il messaggio comprende anche un ID assegnato dal cliente alla coppia corrente richiesta/risposta e un nonce per garantire la tempestività del messaggio.

Il commerciante genera una risposta e la firma con la propria chiave privata di firma. La risposta comprende il nonce proveniente dal cliente, un ulteriore nonce per il cliente da restituire nel messaggio successivo e un ID di transazione per la transazione di acquisto corrente. In aggiunta alla risposta firmata, il messaggio **Initiate Response** comprende il certificato di firma del commerciante e il certificato per lo scambio di chiavi del gateway dei pagamenti.

Il titolare della carta verifica i certificati del commerciante e del gateway dei pagamenti mediante le firme delle rispettive CA e, quindi, crea OI e PI. L'ID di transazione assegnato dal commerciante viene posto sia in OI sia in PI. OI non contiene dati esplicativi sull'ordine quali il numero e il prezzo degli articoli. Piuttosto, contiene un riferimento all'ordine generato nello scambio fra commerciante e cliente durante la fase di acquisto antecedente al primo messaggio SET. Successivamente, il titolare della carta prepara il messaggio **Purchase Request** (Figura 7.10). A tale scopo, il titolare genera una chiave di cifratura simmetrica one-time, K_p . Il messaggio comprende le seguenti informazioni.

1. **Informazioni relative all'acquisto.** Queste informazioni saranno inoltrate al gateway dei pagamenti dal commerciante e comprendono:

- PI;
- doppia firma, calcolata su PI e OI, firmata con la chiave privata di firma del cliente;
- digest del messaggio OI (OIMD).

OIMD è necessario al gateway dei pagamenti per verificare la doppia firma, come visto in precedenza. Tutti questi elementi sono cifrati con K_p . Il risultato finale è il seguente.

- L'involucro digitale. Viene creato cifrando K_p con la chiave pubblica di scambio di chiavi del gateway dei pagamenti. Si chiama involucro digitale in quanto deve essere aperto (decifrato) prima di poter leggere gli altri elementi sopra elencati.

Il valore di K_p non è reso disponibile al commerciante. Pertanto, il commerciante non può leggere alcuna informazione relativa al pagamento.

2. **Informazioni relative all'ordine.** Queste informazioni sono necessarie al commerciante e sono costituite da:

- OI;
- doppia firma, calcolata su PI e OI, firmata con la chiave privata di firma del cliente;
- digest del messaggio PI (PIMD).

PIMD è necessario al commerciante per verificare la doppia firma. Si noti che OI è inviato in chiaro.

3. **Certificato del titolare della carta.** Contiene la chiave pubblica di firma del titolare della carta. È necessario al commerciante e al gateway dei pagamenti.

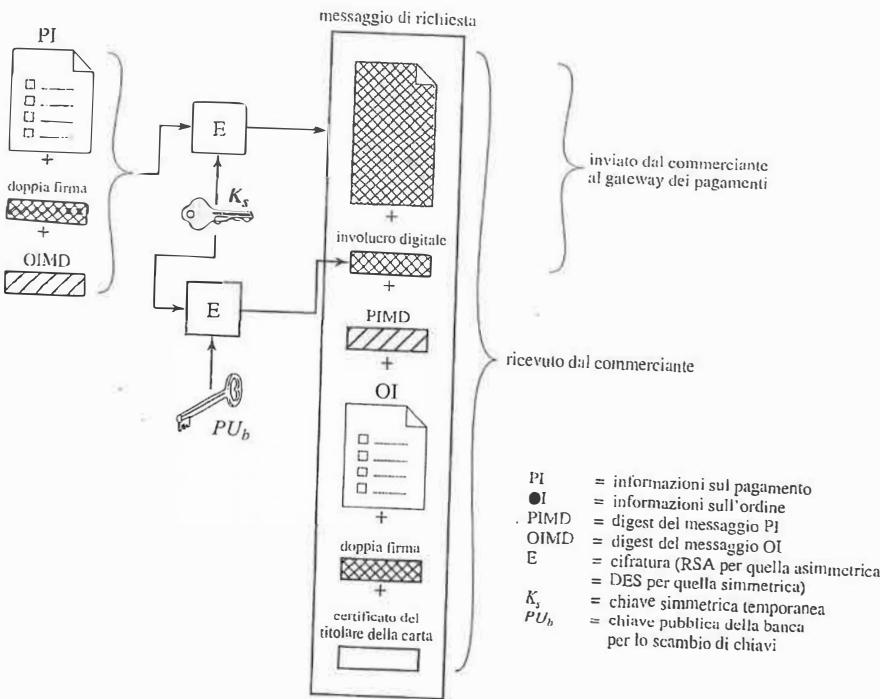


Figura 7.10 Il titolare della carta di credito invia un messaggio Purchase Request.

Quando il commerciante riceve il messaggio Purchase Request, esegue le seguenti azioni (Figura 7.11).

1. Verifica i certificati del titolare della carta mediante le firme della CA ivi apposte.
2. Verifica la doppia firma utilizzando la chiave pubblica di firma del cliente. Questo assicura che l'ordine non è stato manomesso durante la trasmissione e che è stato firmato con la chiave privata di firma del titolare.
3. Elabora l'ordine e inoltra l'informazione relativa al pagamento al gateway dei pagamenti per l'autorizzazione (descritta successivamente).
4. Invia un messaggio Purchase Response al titolare della carta.

Il messaggio Purchase Response comprende un blocco di risposta che conferma l'avvenuta ricezione dell'ordine e fa riferimento al corrispondente identificatore di transazione. Questo blocco è firmato dal commerciante con la sua chiave privata di firma. Il blocco e la sua firma sono inviati al cliente, insieme con il certificato di firma del commerciante.

Quando il software del titolare della carta riceve il messaggio Purchase Response, verifica il certificato del commerciante e, quindi, la firma sul blocco di risposta. Infine, effettua

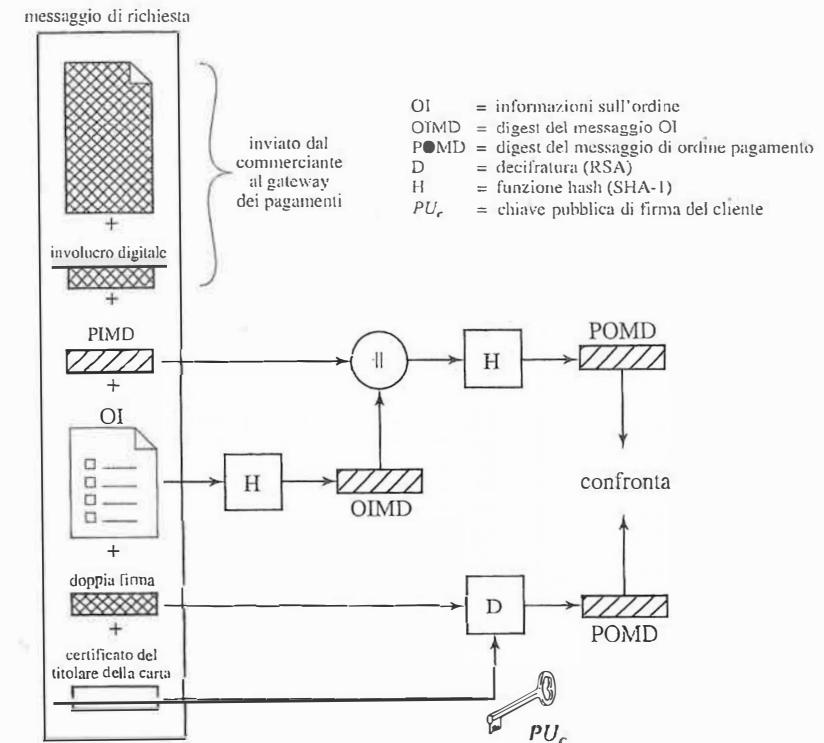


Figura 7.11 Il commerciante verifica il messaggio Purchase Request del cliente.

tua alcune azioni quali mostrare un messaggio all'utente o aggiornare una base di dati con lo stato dell'ordine.

Autorizzazione di pagamento

Durante l'elaborazione di un ordine emesso da un titolare di carta, il commerciante autorizza la transazione con il gateway dei pagamenti. L'autorizzazione di pagamento assicura che la transazione è stata approvata dall'istituto emittente. Questa autorizzazione garantisce che il commerciante riceverà il pagamento; il commerciante può pertanto fornire i servizi o i prodotti al cliente. Lo scambio relativo all'autorizzazione di pagamento è composto da due messaggi:

- Authorization Request (richiesta di autorizzazione)
- Authorization Response (risposta di autorizzazione).

Il commerciante invia un messaggio Authorization Request al gateway dei pagamenti contenente le seguenti informazioni:

1. **Informazioni relative all'acquisto.** Sono state ottenute dal cliente e sono costituite da:
 - PI;
 - doppia firma, calcolata su PI e OI, firmata con la chiave privata di firma del cliente;
 - digest del messaggio OI (OIMD);
 - involucro digitale.
2. **Informazioni relative all'autorizzazione.** Sono generate dal commerciante e sono costituite da:
 - blocco di autorizzazione che comprende l'identificatore di transazione, firmato con la chiave privata di firma del commerciante e cifrato con la chiave simmetrica one-time generata dal commerciante;
 - involucro digitale, costituito cifrando la chiave one-time con la chiave pubblica di scambio di chiavi del gateway dei pagamenti.
3. **Certificati.** Il commerciante include il certificato della chiave di firma del titolare della carta (utilizzato per la verifica della doppia firma), il certificato della chiave di firma del commerciante (utilizzato per la verifica della firma del commerciante), e il certificato di scambio di chiavi del commerciante (necessario nella risposta del gateway dei pagamenti).

Il gateway dei pagamenti esegue le seguenti azioni.

1. Verifica tutti i certificati.
2. Decifra l'involucro digitale del blocco di autorizzazione per ottenere la chiave simmetrica e poi decifra il blocco di autorizzazione stesso.
3. Verifica la firma del commerciante sul blocco di autorizzazione.
4. Decifra l'involucro digitale del blocco di pagamento per ottenere la chiave simmetrica e poi decifra il blocco di pagamento stesso.
5. Verifica la doppia firma sul blocco di pagamento.
6. Verifica che l'identificatore di transazione ricevuto dal commerciante corrisponda a quello nel PI ricevuto (indirettamente) dal cliente.
7. Richiede e riceve un'autorizzazione dall'istituto emittente.

Ottenuta l'autorizzazione dall'istituto emittente, il gateway dei pagamenti restituisce al commerciante un messaggio **Authorization Response**. Tale messaggio comprende i seguenti elementi.

1. **Informazioni relative all'autorizzazione,** che comprendono il blocco di autorizzazione, firmato con la chiave privata di firma del gateway e cifrato mediante una chiave simmetrica one-time generata dal gateway stesso. Le informazioni relative all'autorizzazione comprendono anche un involucro digitale contenente la chiave one-time cifrata con la chiave pubblica di scambio di chiavi del commerciante.

2. **Informazioni relative al capture token,** che saranno utilizzate successivamente per effettuare il pagamento. Questo blocco ha lo stesso formato di (1) ovvero un capture token firmato e cifrato insieme con un involucro digitale. Questo token non viene elaborato dal commerciante ma deve essere restituito così com'è, insieme con la richiesta di pagamento.

3. **Certificato.** Il certificato della chiave di firma del gateway. Ricevuta l'autorizzazione dal gateway, il commerciante può fornire i prodotti o il servizio al cliente.

Riscossione del pagamento

Per ottenere il pagamento, il commerciante impegna il gateway dei pagamenti in una transazione di riscossione del pagamento, che consiste nei messaggi:

- Capture Request (richiesta di riscossione)
- Capture Response (risposta di riscossione).

Per il messaggio **Capture Request**, il commerciante genera, firma e cifra un blocco di richiesta di riscossione, contenente l'importo del pagamento e l'identificatore della transazione. Il messaggio comprende anche il capture token ricevuto precedentemente (nel messaggio Authorization Response) per la transazione corrente, la chiave di firma del commerciante e i certificati delle chiavi di firma e di scambio di chiavi.

Quando il gateway dei pagamenti riceve il messaggio di Capture Request, lo decifra e verifica il blocco di richiesta di riscossione, e decifra e verifica il blocco del capture token. Verifica, quindi, la consistenza fra la richiesta di riscossione e il token di riscossione. Crea, quindi, una richiesta di compenso che viene inviata all'istituto emittente sulla rete privata di pagamento. Questa richiesta consente il trasferimento del denaro sul conto del commerciante.

Il gateway a questo punto notifica il pagamento al commerciante con un messaggio **Capture Response**. Il messaggio comprende un blocco di risposta della riscossione che il gateway firma e cifra. Il messaggio comprende anche il certificato della chiave di firma del gateway. Il software del commerciante memorizza il messaggio di risposta della riscossione allo scopo di utilizzarlo per il ricongiungimento con il pagamento ricevuto dall'istituto acquirente.

7.4 Letture consigliate e siti web

[RESC01] è una buona e dettagliata trattazione di SSL e TLS.

La panoramica più dettagliata di SET è nel primo libro della specifica, disponibile al sito web di SET di MasterCard. Un'altra eccellente rassegna è [MACG97]. Pure [DREW99] è una buona fonte.

Siti web consigliati

- **Pagina SSL di Netscape:** contiene la specifica SSL.
- **Transport Layer Security Charter:** i più recenti documenti RFC e Internet draft relativi a TLS.
- **Progetto OpenSSL:** progetto per sviluppare un software open-source di SSL e TLS. Il sito include documenti e link.

7.5 Domande di revisione ed esercizi

Domande di revisione

- 7.1 Quali sono i vantaggi di ciascuno dei tre approcci illustrati nella Figura 7.1?
- 7.2 Da quali protocolli è costituito SSL?
- 7.3 Qual è la differenza tra una connessione SSL e una sessione SSL?
- 7.4 Elencare e definire brevemente i parametri che definiscono lo stato di una sessione SSL.
- 7.5 Elencare e definire brevemente i parametri che definiscono la connessione di una sessione SSL.
- 7.6 Quali servizi sono forniti dal protocollo Record di SSL?
- 7.7 Quali passi sono richiesti nella trasmissione del protocollo Record di SSL?
- 7.8 Elencare e definire brevemente le principali categorie dei partecipanti SET.
- 7.9 Che cos'è una doppia firma e qual è il suo scopo?

Esercizi

- 7.1 Perché in SSL e TLS si trova un protocollo Change cipher spec separato, e non viene invece incluso un messaggio change_cipher_spec nel protocollo Handshake?
- 7.2 Considerate le seguenti minacce alla sicurezza web e descrivete come ciascuna di queste viene contrastata da una particolare caratteristica di SSL.
 - a. Attacco di crittoanalisi a forza bruta: una ricerca esaustiva nello spazio delle chiavi di un algoritmo di cifratura convenzionale.
 - b. Attacco in cui si conosce il dizionario del testo in chiaro: molti messaggi conterranno testo in chiaro prevedibile come, ad esempio, il comando HTTP GET. Un avversario che perpetra un attacco di questo tipo costruisce un dizionario che contiene ogni possibile cifratura dei messaggi con testo in chiaro conosciuto. Quando viene intercettato un mes-

saggio cifrato, l'avversario prende la porzione con il testo cifrato che corrisponde al testo in chiaro conosciuto e cerca il testo cifrato nel dizionario. Il testo cifrato dovrebbe corrispondere a una voce cifrata con la stessa chiave segreta. Se esistono diverse corrispondenze, si può determinare quella corretta confrontando ciascuna di queste con l'intero testo cifrato. Questo attacco è particolarmente efficace contro chiavi di piccole dimensioni (per esempio, chiavi a 40 bit).

- c. Attacco di replay: viene effettuato un replay ai precedenti messaggi handshake di SSL.
- d. Attacco di tipo man-in-the-middle: durante lo scambio delle chiavi si frappone un avversario che perpetra un attacco che agisce da client per il server e da server per il client.
- e. Intercettazione di password: si intercettano password nel traffico HTTP o di altre applicazioni.
- f. IP Spoofing: si utilizzano indirizzi IP falsificati per ingannare un host inducendolo ad accettare dati falsi.
- g. IP Hijacking (dirottamento IP): una connessione arriva e autenticata fra due host viene interrotta e l'avversario che perpetra l'attacco si sostituisce a uno dei due host.
- h. SYN Flooding (inondazione di messaggi SYN): l'utente che perpetra l'attacco invia messaggi TCP SYN per richiedere una connessione senza rispondere all'ultimo messaggio per stabilire la connessione con successo. Generalmente, il modulo TCP attaccato lascia la connessione aperta a metà (*half-open connection*) per pochi minuti. I messaggi SYN ripetuti possono intasare il modulo TCP.

- 7.3 Sulla base di quanto appreso in questo capitolo, in SSL è possibile per il ricevente riordinare i blocchi dei record SSL che arrivano non rispettando l'ordinamento? In caso affermativo, spieghate come ciò può essere eseguito; altrimenti motivatene l'impossibilità.

Capitolo 8

Sicurezza della gestione di rete

Le reti e i sistemi distribuiti ricoprono oggi un ruolo chiave sempre più rilevante nelle istituzioni pubbliche e private. La linea di tendenza da parte delle organizzazioni è quella di dotarsi di reti sempre più complesse, vaste e articolate, accessibili da un numero sempre maggiore di utenti e applicazioni. L'incremento delle dimensioni delle reti evidenzia inequivocabilmente due problematiche.

- La rete, le risorse ad essa associate e le applicazioni distribuite ricoprono un ruolo di vitale importanza all'interno delle organizzazioni.
- In tale contesto, aumenta la probabilità di malfunzionamenti, che possono rendere inutilizzabile – *in toto* o in parte – una rete o diminuirne le prestazioni a un livello non accettabile.

Una rete di elevate dimensioni non può essere realizzata e gestita solo tramite l'intervento umano. Infatti, l'intrinseca complessità di tali sistemi impone l'utilizzo di strumenti automatizzati per la gestione della rete. L'esigenza di questi strumenti è ancora più sentita nel caso in cui la rete contenga dispositivi di differenti case produttrici. Tale eterogeneità aumenta anche la difficoltà nella realizzazione di strumenti adeguati alla loro gestione. Per fronteggiare queste esigenze, sono stati sviluppati standard che riguardano i servizi, i protocolli e le basi informative di gestione di rete. Allo stato attuale, lo standard maggiormente utilizzato è il protocollo SNMP (*simple network management protocol*, protocollo semplice per la gestione di rete).

Dal momento della sua pubblicazione, avvenuta nel 1988, SNMP è stato utilizzato da un numero sempre crescente di reti e in ambienti sempre più complessi. All'aumentare della diffusione di SNMP, si manifestò la necessità di incorporare nel protocollo nuove funzionalità in grado di soddisfare ulteriori esigenze non previste. Inoltre, diventò evidente come la gestione delle reti non potesse prescindere dal fornire servizi di sicurezza. Per migliorare le funzionalità del protocollo, fu sviluppata la versione 2 di SNMP¹. I miglioramenti rispetto alla sicurezza furono incorporati in SNMPv3.

In questo capitolo saranno descritte sia le preliminari funzionalità di sicurezza previste in SNMPv1, sia quelle più articolate di SNMPv3.

¹ La versione 2 è indicata come SNMPv2. Per distinguerla da quella nuova, la versione originaria è quindi indicata come SNMPv1.

8.1 Concetti base di SNMP

In questo paragrafo si fornirà una panoramica della struttura base di SNMP.

Architettura di gestione di rete

Un sistema per la gestione di rete è una raccolta integrata di strumenti per il monitoraggio e il controllo delle reti. L'integrazione degli strumenti si concretizza nel seguente modo.

- Si può disporre di un'unica interfaccia utente, dotata di un insieme di comandi potenti ma al tempo stesso facili da usare, attraverso cui è possibile effettuare la maggior parte o tutte le attività di gestione di rete.
- È necessario un insieme minimo di apparecchiature aggiuntive in quanto la maggior parte dei dispositivi hardware e software necessari per la gestione di rete è incorporata all'interno dei sistemi utente già esistenti.

Un sistema per la gestione di rete è composto da un insieme di strumenti hardware e software realizzati in modo incrementale sulla base delle componenti di rete già esistenti. Il software utilizzato per realizzare la gestione della rete è contenuto negli elaboratori host e nei processori di comunicazione (per esempio: processori front-end, controllori di cluster di terminali). Un sistema per la gestione di rete è progettato per vedere l'intera rete come un'architettura unificata in cui ogni punto ha associato un indirizzo e un'etichetta, e in cui il sistema di gestione conosce le caratteristiche specifiche di ogni entità o collegamento. Gli elementi che svolgono un ruolo attivo all'interno della rete forniscono regolarmente informazioni di feedback sullo stato al centro di controllo della rete.

Il modello di gestione di rete alla base di SNMP contiene i seguenti elementi chiave:

- stazione di gestione;
- agente gestore
- base informativa di gestione
- protocollo di gestione di rete.

La **stazione di gestione** è generalmente un dispositivo stand-alone, ma può anche essere una funzionalità realizzata nell'ambito di un sistema condiviso. In entrambi i casi, la stazione di gestione costituisce l'interfaccia tra l'utente preposto alla gestione della rete e il sistema di gestione di rete. Il requisito minimo per una stazione di gestione è quello di poter come minimo disporre di:

- un insieme di applicazioni di gestione che attuino operazioni quali l'analisi dei dati e il ripristino da situazioni di errore;
- un'interfaccia che consenta al gestore di rete di monitorare e controllare la rete stessa;
- la capacità di tradurre le esigenze del gestore di rete in operazioni di controllo e monitoraggio degli elementi remoti all'interno della rete;
- una base di dati di informazioni estratte dalle basi informative di gestione presenti in tutte le entità gestite sulla rete.

Solo gli ultimi due elementi sono oggetto di standardizzazione in SNMP.

L'altra entità attiva all'interno di un sistema di gestione di rete è l'**agente gestore**. Elementi chiave, quali host, bridge, router e hub, possono essere equipaggiati con SNMP in modo da poter essere gestiti da una stazione di gestione. L'agente gestore ha il compito di rispondere a richieste di informazioni e a richieste di specifiche azioni provenienti da una stazione di gestione. Inoltre, l'agente gestore può fornire, in modo asincrono, alla stazione di gestione informazioni di notevole rilevanza senza che gli siano state chieste esplicitamente.

Al fine di gestire le risorse di una rete, ogni risorsa è rappresentata tramite un oggetto. In ultima analisi, un oggetto è una variabile dati che rappresenta uno degli aspetti dell'agente gestito. La raccolta di oggetti è denominata **base informativa di gestione** (MIB, *management information base*). La MIB fornisce alla stazione di gestione una serie di punti di accesso presso il corrispondente agente. Tali oggetti sono standardizzati nel caso di sistemi appartenenti a una data classe (per esempio, tutti i bridge forniscono supporto per gli stessi oggetti di gestione). Le attività di controllo sono effettuate da una stazione di gestione tramite il reperimento dei valori degli oggetti contenuti nella MIB. Tramite la modifica del valore di specifici oggetti, una stazione di gestione può causare l'esecuzione di una determinata azione da parte di un dato agente, oppure cambiare la configurazione di un agente.

La stazione di gestione è collegata agli agenti tramite un **protocollo di gestione di rete**. SNMP rappresenta il protocollo utilizzato per la gestione delle reti TCP/IP. Le principali operazioni di tale protocollo sono:

- **Get**: consente alla stazione di gestione di recuperare il valore degli oggetti presso gli agenti;
- **Set**: consente alla stazione di gestione di modificare i valori degli oggetti presso gli agenti;
- **Notify**: consente a un agente di notificare alla stazione di gestione l'occorrenza di eventi significativi.

Architettura del protocollo di gestione di rete

La specifica di SNMP è stata pubblicata nel 1988 e, a partire da allora, SNMP divenne rapidamente lo standard più utilizzato per la gestione di rete. Attualmente, numerose case produttrici forniscono workstation stand-alone preposte alla gestione di rete basate su SNMP, mentre molti produttori di bridge, router, workstation e PC offrono pacchetti contenenti agenti SNMP che consentono di gestire i prodotti da loro commercializzati mediante una stazione di gestione SNMP.

SNMP definisce una MIB limitata ma facile da implementare, contenente variabili numeriche e tabelle bidimensionali. In aggiunta, SNMP definisce un protocollo efficiente che consente al gestore di rete di effettuare operazioni set e get sulle variabili della MIB, e all'agente di inviare notifiche non richieste esplicitamente, denominate **trap**. La semplicità costituisce il punto di forza di SNMP. SNMP è facile da implementare e utilizza una modesta quantità di risorse di elaborazione e di rete. Inoltre, la struttura del protocollo e la MIB sono sufficientemente semplici da rendere agevole l'interoperabilità tra stazioni di gestione ed agenti software forniti da differenti case produttrici.

Le tre specifiche base sono le seguenti.

- **Struttura e riconoscimento delle informazioni di gestione per reti basate su TCP/IP (RFC 1155)**: documento che descrive come sono specificati gli oggetti gestiti contenuti nella MIB.

- Base informativa di gestione per la gestione di rete di intranet basate su TCP/IP; MIB-II (RFC 1213):** documento che descrive gli oggetti contenuti nella MIB.
- Protocollo semplice per la gestione di rete (RFC 1157):** documento che definisce il protocollo per gestire gli oggetti contenuti nella MIB.

SNMP è stato progettato come un protocollo di livello applicazione facente parte della suite di protocolli TCP/IP, in grado di operare al di sopra di UDP (*user datagram protocol*) definito nella RFC 768. Nel caso di stazioni di gestione stand-alone, esiste un processo gestore che controlla gli accessi alla MIB principale posta nella stazione di gestione e fornisce nel contempo un'interfaccia per il gestore di rete. Il processo gestore effettua la gestione della rete mediante SNMP, che viene implementato al di sopra di UDP, di IP e dei principali protocolli che dipendono dalla rete (per esempio: Ethernet, FDDI, X.25).

SNMP, UDP e IP devono essere implementati anche da ciascun agente. Inoltre, esiste un processo agente che interpreta i messaggi SNMP e controlla la MIB dell'agente. Nel caso di dispositivi agente che forniscono supporto ad altre applicazioni, quali ad esempio FTP, è richiesta la presenza sia di TCP sia di UDP.

La Figura 8.1 illustra il contesto in cui opera SNMP. Una stazione di gestione emette tre differenti tipologie di messaggi SNMP per conto di un'applicazione di gestione: GetRequest, GetNextRequest e SetRequest. I primi due messaggi sono varianti della funzione get. L'agente notifica l'avvenuta ricezione di tutti e tre i tipi di messaggio mediante il

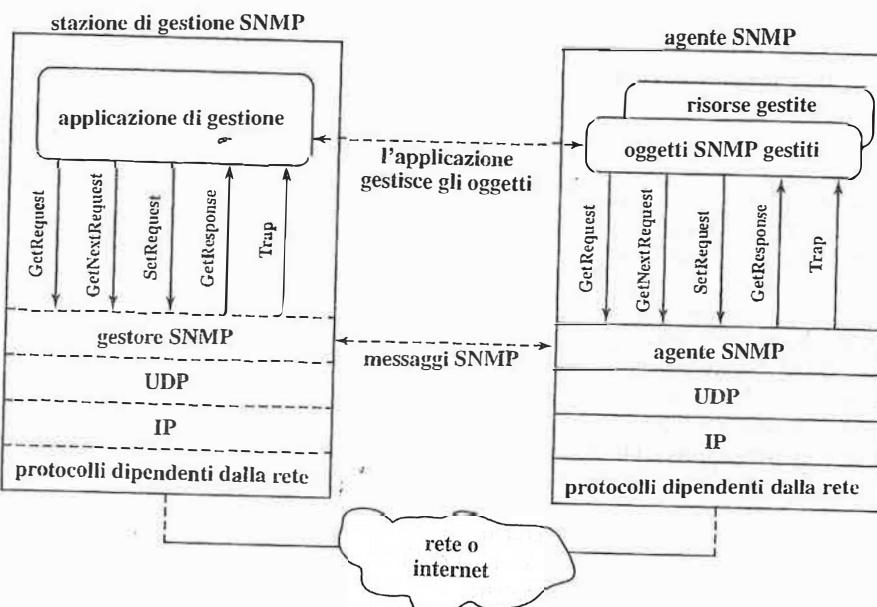


Figura 8.1 Ruolo di SNMP.

messaggio GetResponse, che viene inoltrato all'applicazione di gestione. Inoltre, un agente può emettere un messaggio trap in risposta a un evento che influenza la MIB e le sottostanti risorse gestite.

SNMP è un protocollo privo di connessione (*connectionless*) in quanto si basa su UDP che è, a sua volta, un protocollo connectionless. Non viene quindi mantenuta aperta alcuna connessione tra la stazione di gestione e i suoi agenti. Al contrario, ogni scambio è considerato come una transazione distinta tra una stazione di gestione e un agente.

Proxy

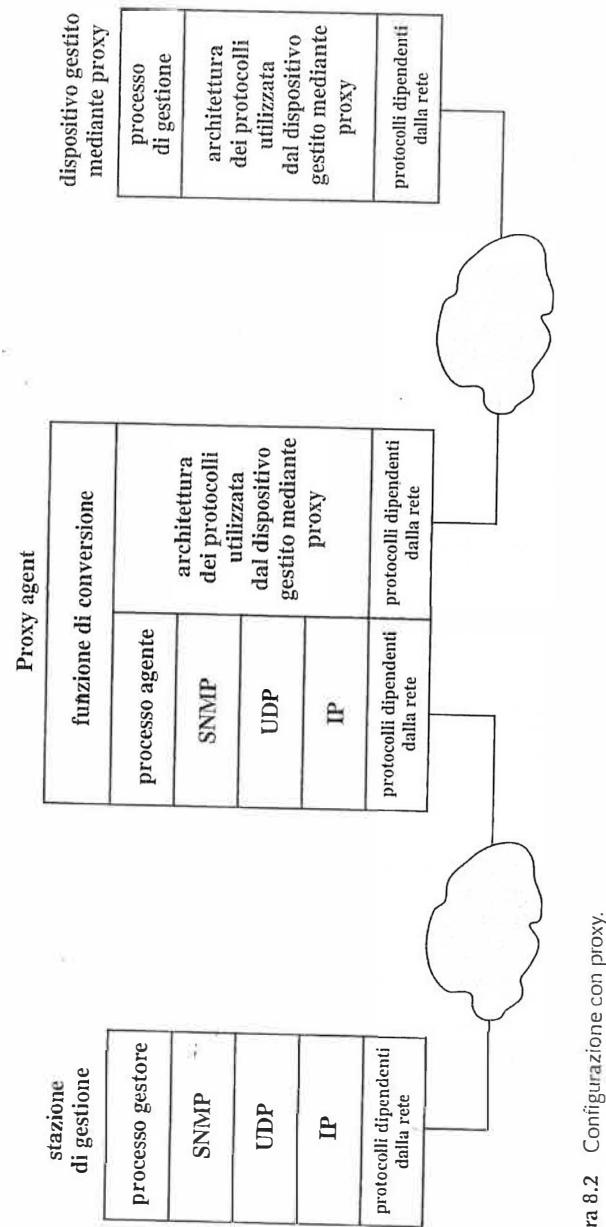
In SNMPv1, sia gli agenti sia le stazioni di gestione devono fornire supporto per UDP e IP. Questo fatto limita la gestione diretta a tali dispositivi e ne esclude altri, quali, ad esempio, alcuni tipi di bridge e di modem che non implementano alcuna parte della suite di protocolli TCP/IP. Inoltre, possono esserci molti sistemi di dimensioni ridotte (personal computer, workstation e controlleri programmabili), che devono implementare TCP/IP per supportare le loro applicazioni, ma per i quali non è conveniente aggiungere l'ulteriore carico connesso a SNMP, alla logica degli agenti e al mantenimento della MIB.

Il concetto di proxy è stato specificamente sviluppato per quei sistemi che non implementano SNMP. In questo caso esiste un agente SNMP che agisce come proxy per uno o più dispositivi; in altri termini, l'agente SNMP agisce per conto dei corrispondenti dispositivi.

La Figura 8.2 illustra lo schema di protocollo più frequentemente utilizzato. La stazione di gestione invia le richieste relative a un dispositivo al suo agente proxy. L'agente proxy traduce ogni richiesta nel protocollo di gestione utilizzato dal dispositivo. Quando l'agente riceve una risposta da parte del dispositivo, la inoltra alla stazione di gestione. Analogamente, se il dispositivo trasmette al proxy la notifica di un certo evento, il proxy inoltra tale notifica alla stazione di gestione sotto forma di trap.

SNMPv2 permette l'utilizzo di protocolli diversi rispetto a quelli contenuti nella suite TCP/IP. Più specificamente, SNMPv2 è stato progettato in modo da poter essere eseguito sulla suite di protocolli OSI. Tale caratteristica rende SNMPv2 adatto alla gestione di una notevole varietà di configurazioni di rete. Tutti i dispositivi che non implementano SNMPv2 possono essere gestiti solo attraverso un proxy. Tale tipologia di gestione si applica anche ai dispositivi che implementano SNMPv1. Questo implica che un dispositivo che realizza un agente software per SNMPv1 può essere raggiunto da un gestore SNMPv2 solo mediante un dispositivo proxy che implementa l'agente SNMPv2 e il software di gestione di SNMPv1.

I casi descritti precedentemente vanno sotto il nome di relazioni con i proxy esterni in SNMPv2, inoltre SNMPv2 supporta le relazioni con i proxy native. SNMPv2 prevede anche il caso in cui dispositivi gestiti mediante proxy implementino SNMPv2. In questo caso, un gestore SNMPv2 comunica con un nodo SNMPv2, che agisce come agente. Questo nodo opera inoltre come un gestore per raggiungere il dispositivo (gestito dal proxy) che agisce come agente SNMPv2. Il motivo per cui è previsto tale schema è quello di consentire agli utenti di configurare sistemi di gestione di rete gerarchici e decentralizzati. Tali aspetti saranno discussi più avanti, nel corso di questo capitolo.



SNMPv2

SNMP fornisce un pacchetto costituito da un insieme base di strumenti per la gestione di rete, facilmente implementabile e configurabile. Per contro, l'utilizzo di SNMP per la gestione di reti in costante espansione e con carichi crescenti, ha evidenziato in modo chiaro i limiti di questo protocollo. Tali limiti sono riassumibili in tre categorie:

- mancanza di supporto per la gestione di rete distribuita;
- limiti funzionali;
- limiti di sicurezza.

I primi due tipi di limiti sono stati superati con SNMPv2, pubblicato nel 1993. Nel 1996 è stata pubblicata una versione rivista, attualmente contenuta nel documento RFC 1901, nei documenti RFC da 1904 a 1908 e nei documenti RFC 2578 e 2579. SNMPv2 è stato immediatamente recepito dal mercato ed entro pochi mesi dalla pubblicazione dello standard erano già disponibili prodotti che lo utilizzavano. I limiti di sicurezza sono stati invece superati con SNMPv3.

Di seguito vedremo in breve le nuove funzionalità fornite da SNMPv2. Le funzionalità di sicurezza di SNMPv1 e di SNMPv3 saranno invece descritte in dettaglio rispettivamente nei Paragrafi 8.2 e 8.3.

Gestione di rete distribuita

In un tradizionale schema centralizzato di gestione di rete, esiste un host che ricopre il ruolo di stazione di gestione; possono inoltre essere presenti una o due stazioni di gestione aggiuntive con il ruolo di stazioni di backup. Tutti gli altri dispositivi presenti sulla rete contengono agenti software e una MIB, per rendere possibile il loro controllo e monitoraggio da parte della stazione di gestione. Tale sistema centralizzato diventa inutilizzabile al crescere delle dimensioni delle reti e del carico di traffico. Il problema è che la stazione di gestione diventa un collo di bottiglia, avendo un carico di lavoro notevole; inoltre, il traffico risulta troppo elevato, in quanto ogni singolo evento che deve essere riferito da un agente deve attraversare l'intera rete per giungere alla stazione di gestione. In tali circostanze, è preferibile adottare un approccio distribuito e decentralizzato (Figura 8.3). In uno schema di gestione di rete decentralizzato, possono essere presenti più stazioni di gestione di alto livello, che possono essere considerate come server di gestione. Ciascuno di questi server può gestire in modo diretto solo una parte dell'insieme totale di agenti. Inoltre, il server di gestione delega la gestione di molti agenti a un gestore intermedio. Quest'ultimo ricopre il ruolo di gestore preposto al controllo e al monitoraggio degli agenti sotto la sua responsabilità. Questi funge anche da agente per fornire informazioni ed essere controllato dai server di gestione di livello più alto. Tale tipo di architettura suddivide il carico di elaborazione e riduce il traffico complessivo di rete.

SNMPv2 fornisce supporto sia per una gestione di rete altamente centralizzata sia per una gestione distribuita. In quest'ultimo caso, esistono alcuni sistemi che agiscono sia da gestore sia da agente. Quando tali sistemi ricoprono il ruolo di agente accettano comandi da un sistema di gestione di livello superiore. Alcuni di questi comandi si riferiscono alla MIB locale presso l'agente. Altri comandi richiedono all'agente di agire come proxy per dispositivi remoti. In altri casi, l'agente proxy assume il ruolo di gestore per accedere a informa-

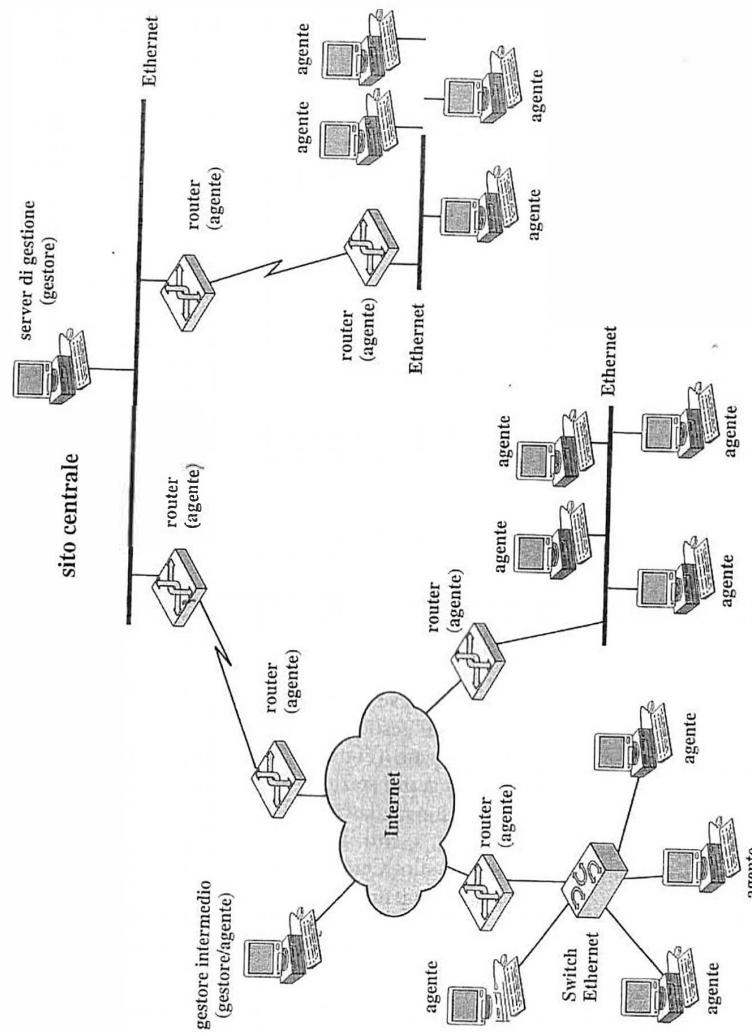


Figura 8.3 Esempio di configurazione della gestione di rete distribuita.

zioni presso un agente remoto e successivamente assume il ruolo di agente per trasferire tali informazioni a un gestore di livello superiore.

Miglioramenti funzionali

La Tabella 8.1 illustra i miglioramenti funzionali che sono stati apportati in SNMPv2. Entrambi i protocolli (SNMPv1 e SNMPv2) sono definiti tramite un insieme di comandi inviati come PDU (*protocol data unit*, unità dati di protocollo). Nel caso di SNMPv1, i comandi sono cinque. Un gestore impedisce il comando Get a un agente per recuperare i valori degli oggetti contenuti nella MIB. Il comando GetNext sfrutta il fatto che gli oggetti in una MIB sono organizzati in una struttura ad albero. Quando un oggetto compare in un comando GetNext, l'agente ricerca l'oggetto a lui successivo nell'albero e ne restituisce il valore. Il comando GetNext è utile in quanto consente al gestore di "attraversare" la struttura ad albero posta presso un agente, nel caso in cui non sia a conoscenza dell'esatto insieme di oggetti presenti presso quel particolare agente. Il comando Set permette a un gestore di aggiornare i valori presso un agente; inoltre, tale comando è utilizzato per inserire e cancellare righe nelle tabelle. Il comando GetResponse viene utilizzato da un agente per rispondere a un comando di un gestore. Infine, il comando Trap consente a un agente di inviare informazioni a un gestore senza dover aspettare un'esplicita richiesta. Ad esempio, un agente potrebbe essere configurato in modo da inviare un comando Trap quando fallisce un collegamento, oppure quando il traffico supera una soglia prefissata.

In aggiunta a tutti quelli previsti da SNMPv1, SNMPv2 contiene due ulteriori comandi. Il più importante di questi è Inform. Questo comando è inviato da una stazione di gestione

PDU SNMPv1	PDU SNMPv2	Direzione	Descrizione
GetRequest	GetRequest	Da gestore ad agente	Richiesta di valore per ciascun oggetto elencato
GetNextRequest	GetNextRequest	Da gestore ad agente	Richiesta di valore successivo per ciascun oggetto elencato
-	GetBulkRequest	Da gestore ad agente	Richiesta di più valori
SetRequest	SetRequest	Da gestore ad agente	Inizializzazione del valore per ciascun oggetto elencato
-	InformRequest	Da gestore a gestore	Trasmissione di informazione non richiesta
GetResponse	Response	Da agente a gestore o da gestore a gestore (SNMPv2)	Risposta a una richiesta del gestore
Trap	SNMPv2-Trap	Da agente a gestore	Trasmissione di informazione non richiesta

Tabella 8.1 Confronto delle PDU di SNMPv1 e SNMPv2.

a un'altra e, come nel caso di un comando Trap, contiene informazioni che sono correlate a condizioni o eventi presso colui che invia il comando. Il vantaggio del comando Inform è che può essere utilizzato per predisporre una configurazione in cui esistono molteplici gestori che cooperano per condividere la responsabilità della gestione di una rete di dimensioni elevate.

L'altro comando nuovo è GetBulk che consente a un gestore di recuperare in una sola volta un grosso blocco di dati. Più specificatamente, il comando GetBulk è stato concepito per consentire la trasmissione di intere tabelle mediante un solo comando.

Un'ultima differenza tra i due protocolli consiste nel fatto che il comando Get è atomico in SNMPv1, mentre non ha questa proprietà in SNMPv2. Quindi, se il comando Get di SNMPv1 contiene una lista di oggetti di cui sono richiesti i valori, ed anche uno solo di questi non è presente presso l'agente, l'intero comando non verrà eseguito. Nel caso di SNMPv2, è invece possibile restituire risultati parziali. La versione non atomica del comando Get consente al gestore di utilizzare in modo più efficiente la capacità di rete.

8.2 Funzionalità di comunità di SNMPv1

La definizione di SNMPv1 contenuta nel documento RFC 1157 fornisce solo una rudimentale funzionalità di sicurezza basata sul concetto di comunità. Questa funzionalità offre un certo livello di sicurezza, ma è aperta a diversi attacchi [CERT02, JIAN02].

Comunità e nomi di comunità

Come altre applicazioni distribuite, anche la gestione di rete implica l'interazione tra un certo numero di entità applicative mediante l'utilizzo di un protocollo a livello applicazione. Nel caso della gestione di rete SNMP, le entità applicative sono le applicazioni gestore e le applicazioni agente che utilizzano SNMP.

La gestione di rete SNMP ha alcune caratteristiche peculiari, non riscontrabili in tutte le applicazioni distribuite. Il protocollo implica una relazione uno a molti tra il gestore e un insieme di agenti: il gestore è in grado di effettuare operazioni get e set sugli oggetti presso gli agenti ed è anche in grado di ricevere trap dagli agenti. In questo modo, da un punto di vista operazionale o di controllo, il gestore "gestisce" un certo numero di agenti. Ci può anche essere più di un gestore, ciascuno dei quali gestisce tutti – o solo una parte – degli agenti previsti dalla configurazione. I sottoinsiemi di agenti gestiti dai vari gestori non devono essere necessariamente disgiunti.

La gestione di rete SNMP può essere vista, in modo alternativo, come una relazione uno a molti tra un agente e un insieme di gestori. Infatti, ciascun agente controlla la propria MIB locale e deve essere in grado di controllare l'utilizzo di tale MIB da parte di più gestori. In questo tipo di controllo, sono presenti tre diversi aspetti.

- **Servizio di autenticazione:** l'agente può voler limitare gli accessi alla sua MIB solo a un insieme di gestori autorizzati.
- **Politica di accesso:** l'agente può voler concedere differenti tipologie di accesso a diversi gestori.

- **Servizio proxy:** un agente può operare come proxy per altri agenti. Tale servizio può implicare l'implementazione di un servizio di autenticazione e/o di una politica di accesso per gli altri agenti nel sistema proxy.

Tutti questi aspetti comportano problematiche di sicurezza. In un ambiente che prevede la suddivisione delle responsabilità per quanto riguarda le componenti di rete – in analogia con quanto avviene, ad esempio, nel caso di un insieme di entità amministrative – gli agenti hanno bisogno di proteggere se stessi e le proprie MIB da accessi indesiderati e non autorizzati. La definizione di SNMP contenuta nel documento RFC 1157 fornisce solo una limitata e primitiva possibilità di proteggersi da questi tipi di accesso, realizzata tramite il concetto di comunità.

Una **comunità SNMP** è una relazione tra un agente SNMP e un insieme di gestori SNMP che definisce le caratteristiche dell'autenticazione, del controllo dell'accesso e dei proxy. Il concetto di comunità è un concetto locale, definito a livello di agente. L'agente istituisce una comunità per ogni combinazione di caratteristiche di autenticazione, controllo dell'accesso e proxy che si vogliono ottenere. A ciascuna comunità è assegnato un nome univoco, per il particolare agente considerato, che viene fornito a tutti i gestori all'interno della comunità. I gestori devono utilizzare tale nome per tutte le operazioni get e set che vogliono effettuare presso quell'agente. Un agente può istituire più comunità, e un gestore può appartenere a più comunità istituite dallo stesso agente.

Dal momento che le comunità sono definite localmente presso l'agente, lo stesso nome di comunità può essere utilizzato da agenti differenti. Il fatto che due comunità abbiano lo stesso nome è irrilevante e non è sintomatico di nessuna correlazione tra le due. Un gestore, quindi, deve tenere traccia dei nomi di tutte le comunità corrispondenti agli agenti cui vuole accedere.

Servizio di autenticazione

Lo scopo del servizio di autenticazione di SNMPv1 è assicurare chi riceve un messaggio SNMPv1 del fatto che tale messaggio provenga proprio dalla sorgente dichiarata. Lo schema di autenticazione previsto da SNMPv1 è banale. Ogni messaggio inviato da un gestore a un agente contiene un nome di comunità. Questo nome ha la stessa funzione di una password, e il messaggio è considerato autentico se il mittente conosce la password.

Dato che la forma di autenticazione è alquanto limitata, molti gestori di rete sono riluttanti a permettere operazioni che vadano al di là del monitoraggio di rete, ovvero operazioni get e trap. Il controllo della rete tramite operazioni di set è chiaramente un'area più delicata. Il nome della comunità può essere usato per far partire una procedura di autenticazione e tale nome viene usato come password. La procedura di autenticazione può utilizzare tecniche di cifratura/decifratura per aumentare la sicurezza. Tale tipo di funzionalità va però oltre lo scopo del documento RFC 1157.

Politica di accesso

Istituendo una comunità, un agente limita gli accessi alla sua MIB solo a un insieme selezionato di gestori. Utilizzando più di una comunità, l'agente può consentire a gestori diversi differenti classi di accesso alla MIB. Due sono gli aspetti connessi a questa tipologia di controllo dell'accesso.

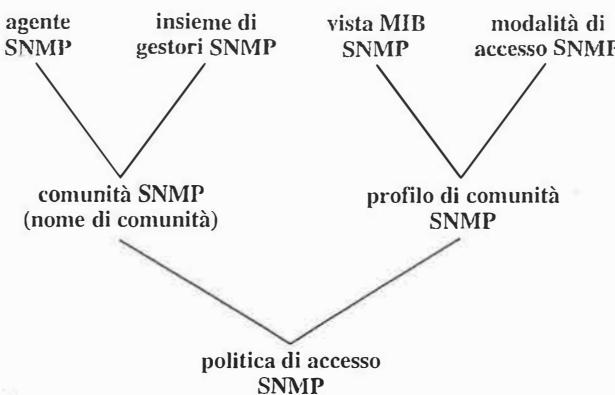


Figura 8.4 Elementi di amministrazione di SNMPv1.

- Vista MIB SNMP:** è un sottoinsieme degli oggetti contenuti in una MIB. Per ciascuna comunità è possibile definire una vista MIB diversa. Non è necessario che gli oggetti che compongono una vista appartengano tutti allo stesso sottoalbero della MIB.
- Modalità di accesso SNMP:** è un elemento dell'insieme {READ-ONLY, READ-WRITE}. Per ciascuna comunità viene definita la modalità di accesso opportuna.

La coppia costituita da una certa vista MIB e da una data modalità di accesso è denominata **profilo di comunità SNMP**. Un profilo di comunità è quindi costituito da un sottoinsieme degli oggetti contenuti nella MIB di un agente e da una modalità di accesso per tali oggetti. La modalità di accesso SNMP è applicata senza distinzioni a tutti gli oggetti contenuti nella vista in questione. Ad esempio, se viene scelta la modalità di accesso READ-ONLY, tale modalità di accesso si applica a tutti gli oggetti della vista, e limita l'accesso dei gestori a questa vista alle sole operazioni di lettura.

Un profilo di comunità viene associato a tutte le comunità definite da un agente: la combinazione di una comunità SNMP con un profilo viene denominata **politica di accesso SNMP**. La Figura 8.4 illustra i concetti appena introdotti.

Servizio proxy

Il concetto di comunità è utile anche per fornire supporto al servizio proxy. Si ricordi che un proxy è un agente SNMP che agisce per conto di altri dispositivi. In genere, tali dispositivi sono esterni, nel senso che non implementano TCP/IP e SNMP. In alcuni casi, il sistema gestito tramite proxy può fornire supporto per SNMP. In questo caso il proxy è utilizzato per minimizzare l'interazione tra il dispositivo e i sistemi di gestione di rete.

Il sistema proxy mantiene una politica di accesso SNMP per ciascun dispositivo che gestisce. Il proxy conosce quindi quali oggetti nella MIB (cioè quelli che compongono la vista MIB) possono essere utilizzati per gestire il dispositivo in questione e la loro modalità di accesso.

8.3 SNMPv3

Nel 1998, il gruppo di lavoro IETF relativo a SNMPv3 realizzò un insieme di Proposed standard, attualmente raccolti nei documenti RFC da 2570 a 2575. Questo insieme di documenti definisce un riferimento per incorporare funzionalità di sicurezza all'interno di un'architettura che include le funzionalità SNMPv1 o SNMPv2. I documenti definiscono inoltre un insieme specifico di funzionalità per la sicurezza di rete e il controllo dell'accesso.

È importante comprendere come SNMPv3 non costituisca solo un'alternativa stand-alone che sostituisce SNMPv1 e/o SNMPv2. Le funzionalità di sicurezza previste da SNMPv3 possono essere utilizzate in combinazione con SNMPv2 (soluzione consigliata) o con SNMPv1. Inoltre, il documento RFC 2571 descrive un'architettura in grado di contemplare tutte le versioni attuali e future di SNMP. Il documento RFC 2575 descrive una funzionalità di controllo dell'accesso, realizzata per operare in modo indipendente rispetto alle funzionalità base di SNMPv3. In questo paragrafo verrà fornita una visione di insieme e una rassegna delle funzionalità definite nei documenti RFC da 2570 a 2576.

La Figura 8.5 illustra le relazioni esistenti tra le varie versioni di SNMP, con riferimento ai formati utilizzati. Le informazioni sono scambiate tra una stazione di gestione e un agente, sotto forma di messaggio SNMP. Le elaborazioni relative alla sicurezza avvengono a livello del messaggio; ad esempio, SNMPv3 definisce un **modello di sicurezza utente** (USM, *user security model*) che fa uso di campi nell'intestazione del messaggio. Il payload di un messaggio SNMP è costituito da una PDU SNMPv1 o SNMPv2. Una PDU denota una tipologia di azione di gestione (per esempio, un'operazione get o set su un oggetto gestito) e una lista di nomi di variabili connesse a questa azione.

I documenti RFC da 2570 a 2576 descrivono un'architettura generale, la struttura di messaggi specifici e le funzionalità di sicurezza, ma non definiscono un nuovo formato per

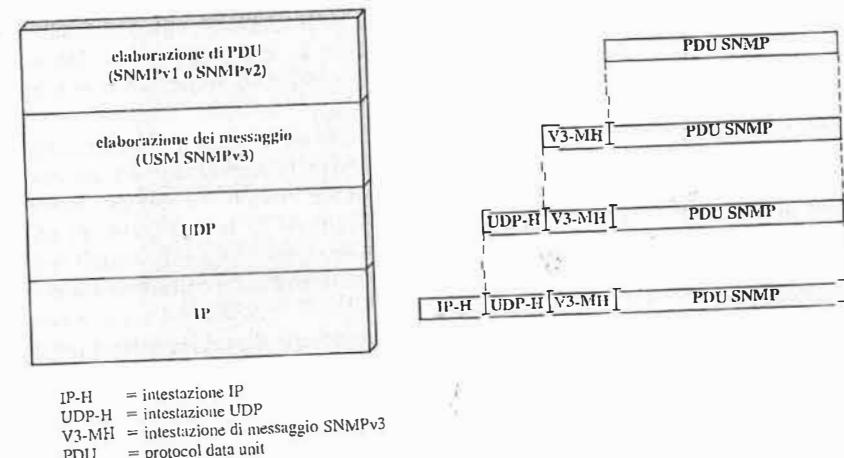


Figura 8.5 Architettura dei protocolli SNMP.

le PDU SNMP. Questo implica che, all'interno di questa nuova architettura, devono essere utilizzati i formati PDU SNMPv1 o SNMPv2 esistenti. Un'implementazione di SNMPv3 consiste nelle funzionalità architettoniche e di sicurezza definite nei documenti RFC da 2570 a 2576 con l'aggiunta del formato delle PDU e delle funzionalità definite nei documenti relativi a SNMPv2. Tale concetto è così espresso dal documento RFC 2570: "SNMPv3 può essere considerato come SNMPv2 con funzionalità aggiuntive per l'amministrazione e la sicurezza".

Architettura SNMP

L'architettura SNMP descritta nel documento RFC 2571 consiste in un insieme distribuito di entità SNMP che interagiscono fra loro. Ogni entità realizza una parte delle funzionalità di SNMP e può operare come nodo agente, nodo gestore o come una combinazione dei due. Ogni entità SNMP è formata da una raccolta di moduli che interagiscono tra loro al fine di fornire servizi. Queste interazioni possono essere modellate tramite un insieme di primitive astratte e di parametri.

L'architettura proposta nel documento RFC 2571 riflette un requisito chiave del progetto di SNMPv3 e cioè quello di realizzare un'architettura modulare che (1) consenta l'implementazione in una vasta gamma di ambienti operazionali, alcuni dei quali richiedono funzionalità minime e poco costose, mentre altri possono fornire supporto per funzionalità aggiuntive per la gestione di reti di grosse dimensioni; (2) renda possibile far progredire alcune parti dell'architettura nell'iter di standardizzazione anche se non si è ancora ottenuto un consenso generale su tutte le parti dell'architettura stessa; (3) fornisca la possibilità di utilizzare modelli di sicurezza alternativi.

Entità SNMP

Ciascuna entità SNMP contiene un singolo motore SNMP (*SNMP engine*). Un motore SNMP realizza le funzioni necessarie per inviare e ricevere messaggi, effettuare l'autenticazione, cifrare/decifrare messaggi e per controllare gli accessi agli oggetti gestiti. Tali funzionalità sono fornite sotto forma di servizi a una o più applicazioni, configurate in modo da formare, insieme al motore SNMP, un'entità SNMP.

Sia il motore SNMP che le relative applicazioni sono definite come una raccolta di moduli distinti. Questa architettura presenta numerosi vantaggi. In primo luogo, il ruolo di un'entità SNMP è determinato dai moduli implementati al suo interno. Ad esempio, la realizzazione di un agente SNMP richiederà l'implementazione di un certo insieme di moduli, mentre quella per un gestore SNMP richiederà un insieme differente di moduli (sebbene con qualche intersezione). In secondo luogo, la struttura modulare della specifica si presta alla definizione di versioni diverse dei vari moduli. Questa caratteristica rende a sua volta possibile (1) definire funzionalità alternative o miglioramenti di quelle esistenti per alcuni aspetti di SNMP, senza la necessità di definire una nuova versione dell'intero standard (per esempio, SNMPv4), e (2) specificare chiaramente le strategie per la coesistenza e la transizione da una versione all'altra (RFC 2576).

Per meglio comprendere il ruolo di ogni modulo e la sua relazione con gli altri, è meglio analizzarne l'utilizzo nei tradizionali gestori ed agenti SNMP. Il termine "tradizionale", sinonimo in questo contesto di "puro", è utilizzato per enfatizzare il fatto che

una data implementazione non deve essere necessariamente un puro gestore o un puro agente, ma può contenere moduli che consentono a un'entità di eseguire i compiti sia di gestore sia di agente.

La Figura 8.6, basata su una figura contenuta nel documento RFC 2571, è un diagramma a blocchi di un **gestore SNMP tradizionale**. Un gestore SNMP tradizionale interagisce con gli agenti SNMP tramite l'invio di comandi (get, set) e la ricezione di messaggi trap; il gestore può anche interagire con altri gestori mediante l'invio di PDU InformRequest, trai contenenti segnali di allarme, e la ricezione di PDU InformResponse, utilizzate per segnalare l'avvenuta ricezione delle PDU InformRequest. Nella terminologia SNMPv3, un gestore SNMP tradizionale comprende tre categorie di applicazioni.

- **Applicazioni di generazione dei comandi:** controllano e manipolano i dati di gestione presenti presso gli agenti remoti; utilizzano le PDU SNMPv1 e/o SNMPv2, inclusi i comandi Get, GetNext, GetBulk e Set.
- **Applicazione di generazione delle notifiche:** ha il compito di generare messaggi asincroni; nel caso di un gestore tradizionale tale applicazione utilizza la PDU InformRequest.

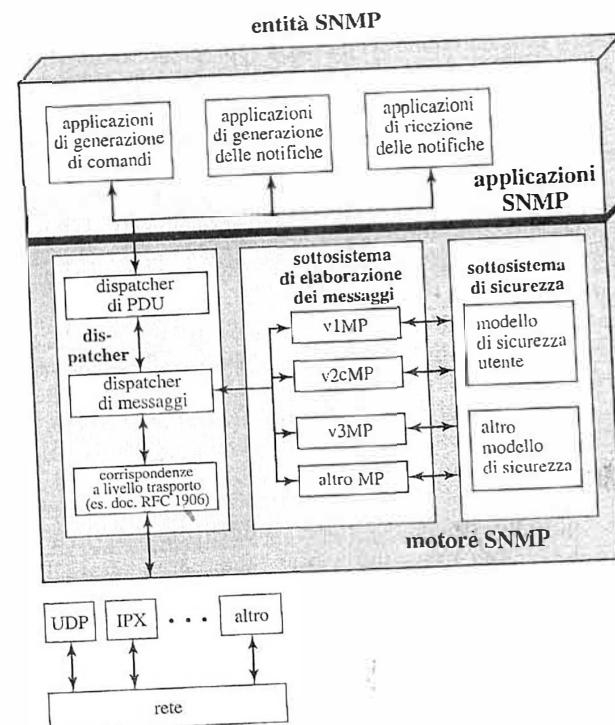


Figura 8.6 Gestore SNMP tradizionale.

- Applicazione di ricezione delle notifiche:** elabora i messaggi asincroni in arrivo che includono PDU InformRequest e trap SNMPv2 e SNMPv1. Nel caso venga ricevuta una PDU InformRequest, l'applicazione di ricezione delle notifiche risponderà mediante una PDU Response.

Tutte le applicazioni descritte in precedenza utilizzano i servizi che il motore SNMP mette loro a disposizione. Il motore SNMP effettua due funzioni di carattere generale.

- Accetta PDU inviate da applicazioni SNMP, effettua le necessarie elaborazioni, inclusi gli inserimenti di codici di autenticazione e la cifratura, e infine incapsula le PDU in opportuni messaggi, per effettuare la loro trasmissione.
- Accetta dal livello trasporto messaggi SNMP in arrivo; effettua le elaborazioni necessarie, incluse operazioni di autenticazione e decifratura; successivamente estrae le PDU dai messaggi e le inoltra alle applicazioni SNMP appropriate.

Nel caso di gestori SNMP tradizionali, il motore SNMP contiene un dispatcher, un sottosistema per l'elaborazione di messaggi e un sottosistema di sicurezza. Il dispatcher è un semplice gestore del traffico. Nel caso di PDU in uscita, il dispatcher accetta le PDU dalle applicazioni e svolge le seguenti funzioni. Per ciascuna PDU, determina il tipo di elaborazione del messaggio richiesta (cioè SNMPv1, SNMPv2c² o SNMPv3) e, sulla base di questo, inoltra la PDU al modulo di elaborazione dei messaggi appropriato, all'interno del sottosistema per l'elaborazione di messaggi. Successivamente, il sottosistema per l'elaborazione di messaggi restituisce un messaggio contenente la PDU in questione e le intestazioni di messaggio appropriate. A questo punto, il dispatcher invia il messaggio a livello trasporto per effettuare la trasmissione.

Per quanto riguarda i messaggi in arrivo, il dispatcher accetta messaggi dal livello trasporto e svolge le seguenti funzioni: instrada ogni messaggio verso l'appropriato modulo di elaborazione; successivamente, il sottosistema per l'elaborazione di messaggi restituisce la PDU contenuta nel messaggio. A questo punto, il dispatcher inoltra questa PDU all'applicazione appropriata.

Il sottosistema per l'elaborazione di messaggi riceve dal dispatcher le PDU in uscita e le prepara per la trasmissione, completandole con l'opportuna intestazione di messaggio e restituendole quindi al dispatcher. Il sottosistema per l'elaborazione di messaggi riceve inoltre dal dispatcher i messaggi in arrivo, elabora le loro intestazioni e restituisce al dispatcher la PDU in essi contenuta. Una realizzazione del sottosistema per l'elaborazione di messaggi può fornire supporto per un unico formato di messaggio che corrisponde a un'unica versione di SNMP (SNMPv1, SNMPv2c o SNMPv3), oppure può contenere un insieme di moduli, ciascuno dei quali fornisce supporto per una differente versione di SNMP.

Il sottosistema di sicurezza svolge funzioni di autenticazione e cifratura. Ogni messaggio in uscita è inoltrato dal sottosistema per l'elaborazione di messaggi al sottosistema di sicurezza. Sulla base dei servizi richiesti, il sottosistema di sicurezza può cifrare la PDU contenuta nel messaggio ed eventualmente anche alcuni campi dell'intestazione del messaggio. Il sottosistema di sicurezza può, inoltre, generare un codice di autenticazione e

inserirlo nell'intestazione del messaggio. Il messaggio così elaborato viene quindi restituito al sottosistema per l'elaborazione di messaggi. In modo analogo, il sottosistema per l'elaborazione di messaggi inoltra al sottosistema di sicurezza ogni messaggio in arrivo. Se richiesto, il sottosistema di sicurezza verifica il codice di autenticazione ed effettua la decifrazione. Il messaggio così elaborato è successivamente restituito al sottosistema per l'elaborazione di messaggi. Ogni implementazione del sottosistema di sicurezza può fornire supporto per uno o più modelli di sicurezza distinti. Allo stato attuale, l'unico modello di sicurezza che è stato definito è il modello di sicurezza utente definito per SNMPv3 e specificato nel documento RFC 2574.

La Figura 8.7, basata su una figura contenuta nel documento RFC 2571, contiene un diagramma a blocchi per un agente SNMP tradizionale. Un agente tradizionale può contenere tre diversi tipi di applicazioni.

- Applicazioni di risposta ai comandi:** forniscono l'accesso ai dati di gestione; rispondono a richieste di reperimento di oggetti e/o a richieste di modifica degli oggetti gestiti. La risposta a tali richieste è una PDU Response.
- Applicazioni di generazione delle notifiche:** generano messaggi asincroni; nel caso di agenti tradizionali, tali applicazioni utilizzano la PDU trap SNMPv2 o SNMPv1.
- Applicazioni proxy per l'inoltro:** hanno il compito di inoltrare messaggi tra le entità.

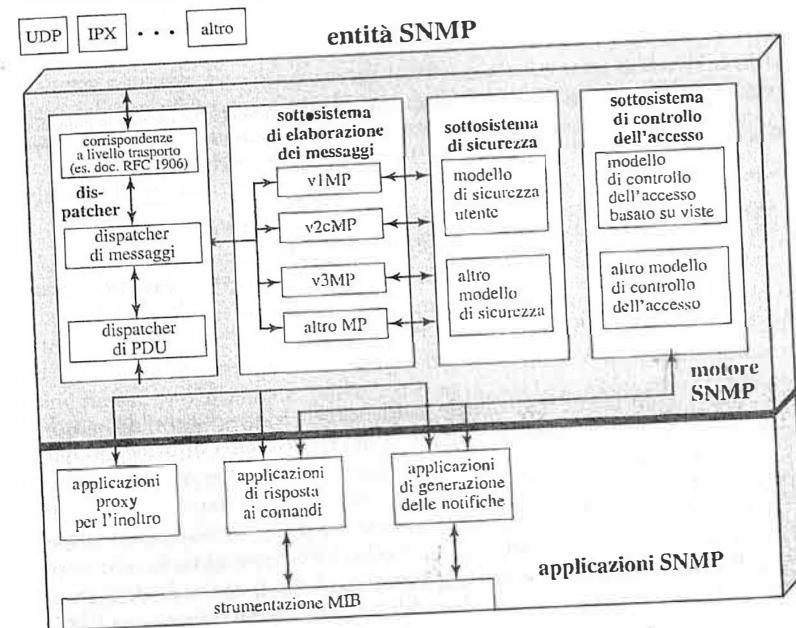


Figura 8.7 Agente SNMP tradizionale.

² SNMPv2 con l'aggiunta del concetto di comunità di SNMPv1.

snmpEngineID

Identificatore univoco e non ambiguo di un motore SNMP e dell'entità SNMP corrispondente al motore in questione. Definito come stringa di byte.

contextEngineID

Identifica univocamente un'entità SNMP che può realizzare un'istanza di contesto con un particolare contextName.

contextName

Identifica un particolare contesto nell'ambito di un motore SNMI. È passato come parametro al dispatcher e al sottosistema di controllo dell'accesso.

scopedPDU

Blocco di dati che comprende: un valore contextEngineID, un valore contextName e una PDU SNMP. È un parametro di ingresso/uscita per il sottosistema di sicurezza.

snmpMessageProcessingModel

Identificatore univoco di un modello di elaborazione dei messaggi del sottosistema per l'elaborazione dei messaggi. Valori possibili comprendono SNMPv1, SNMPv2c, SNMPv3. Definito come intero.

snmpSecurityModel

Identificatore univoco di un modello di sicurezza del sottosistema di sicurezza. Valori possibili sono SNMPv1, SNMPv2c e USM. Definito come intero.

snmpSecurityLevel

Livello di sicurezza cui si può inviare un messaggio SNMP o con il quale si possono effettuare le operazioni. espresso sotto forma di richiesta/non richiesta di autenticazione e/o riservatezza. I valori possibili sono: noAuthNoPriv, authNoPriv e authPriv. Definito come intero.

principal

Entità per conto della quale sono forniti i servizi o viene effettuata l'elaborazione. Può essere: un individuo che agisce nell'ambito di un certo ruolo; un insieme di individui in cui ciascuno agisce nell'ambito di un certo ruolo; un'applicazione o un insieme di applicazioni; una combinazione dei precedenti.

securityName

Stringa comprensibile che denota un principal. Passata come parametro a tutte le primitive SNMP (Dispatcher, Elaborazione del messaggio, Sicurezza, Controllo degli accessi).

Tabella 8.2 Terminologia SNMPv3.

Il motore SNMP per un agente tradizionale contiene tutte le componenti del motore SNMP relativo a un gestore tradizionale, con l'aggiunta del sottosistema di controllo dell'accesso. Questo sottosistema fornisce servizi di autorizzazione per controllare gli accessi alle MIB, effettuati per leggere e modificare gli oggetti in esse contenuti. Tali servizi sono effettuati sulla base del contenuto delle PDU. Una qualsiasi realizzazione del sottosistema di sicurezza può fornire supporto per uno o più modelli di controllo dell'accesso distinti. Allo stato attuale, l'unico modello di sicurezza che è stato definito è il modello di controllo dell'accesso basato su viste (VACM, *view-based access control model*), specificato nel documento RFC 2575.

Si noti come le funzionalità di sicurezza siano organizzate all'interno di due sottosistemi distinti: sicurezza e controllo dell'accesso. Tale organizzazione è un eccellente esempio

di una buona progettazione modulare in quanto i due sottosistemi svolgono funzioni abbastanza diverse, e ha senso pertanto consentire che il processo di standardizzazione per le due aree proceda in maniera indipendente. Il sottosistema di sicurezza si occupa di aspetti di riservatezza ed autenticazione ed agisce sui messaggi SNMP. Il sottosistema di controllo dell'accesso si occupa invece di autorizzare l'accesso alle informazioni relative alla gestione e opera sulle PDU SNMP.

Terminologia

La Tabella 8.2 definisce brevemente alcuni termini introdotti nel documento RFC 2571. A ogni entità SNMP è associato un identificatore univoco chiamato snmpEngineID. Per scopi di controllo dell'accesso, si ipotizza che ogni entità SNMP gestisca un certo numero di contesti di oggetti gestiti, ciascuno dei quali è identificato da un contextName che è unico per ogni entità. Per enfatizzare il fatto che esiste un unico gestore di contesti all'interno di un'entità, ciascuna entità ha associato un singolo identificatore, chiamato contextEngineID; dato che esiste una corrispondenza uno a uno tra il motore di contesto (*context engine*) e il motore SNMP dell'entità corrispondente, il valore del contextEngineID è identico al valore del snmpEngineID. Il controllo dell'accesso è regolato dallo specifico contesto cui si richiede accesso e dall'identità dell'utente che richiede l'accesso; tale identità è espressa sotto forma di principal che può essere un individuo, un'applicazione oppure un gruppo di individui o applicazioni.

Altri termini rilevanti sono quelli connessi all'elaborazione dei messaggi. Il valore di snmpMessageProcessingModel stabilisce il formato del messaggio e la versione di SNMP da utilizzare per l'elaborazione del messaggio. Il valore di snmpSecurityModel specifica quale modello di sicurezza utilizzare. Il valore di snmpSecurityLevel determina quali servizi di sicurezza sono richiesti dalla specifica operazione. L'utente può richiedere la sola autenticazione oppure l'autenticazione congiuntamente alla riservatezza (cifratura), o nessuna delle due.

Applicazioni SNMPv3

I servizi offerti tra i vari moduli che costituiscono un'entità SNMP sono definiti nei documenti RFC tramite una serie di primitive e parametri. Una primitiva specifica la funzione che deve essere eseguita, mentre i parametri sono utilizzati per passare alla funzione dati e informazioni di controllo. Si può pensare a primitive e parametri come a un modo formale per definire i servizi SNMP. La realizzazione effettiva di una primitiva è dipendente dalla specifica realizzazione, un esempio è una chiamata a procedura. A questo proposito, può essere utile fare riferimento alla Figura 8.8, che si basa su una figura contenuta nel documento RFC 2571, per capire come tutte le primitive coesistono insieme. La Figura 8.8a illustra la sequenza di eventi tramite cui un'applicazione di generazione dei comandi o un'applicazione di generazione delle notifiche richiede l'invio di una PDU, e come la relativa risposta sia restituita all'applicazione; questi eventi hanno luogo presso un gestore. La Figura 8.8b mostra i corrispondenti eventi presso un agente. Si può vedere come l'arrivo di un messaggio causi l'invio a un'applicazione della PDU in esso contenuta, e come la risposta di questa applicazione dia origine all'invio di un messaggio. Si noti che alcune delle frecce nel diagramma sono etichettate con il nome di una primitiva per denotare una chiamata. Le frecce non etichettate indicano il ritorno da una chiamata, mentre le zone in grigio rappresentano il collegamento tra una chiamata di procedura e il ritorno:

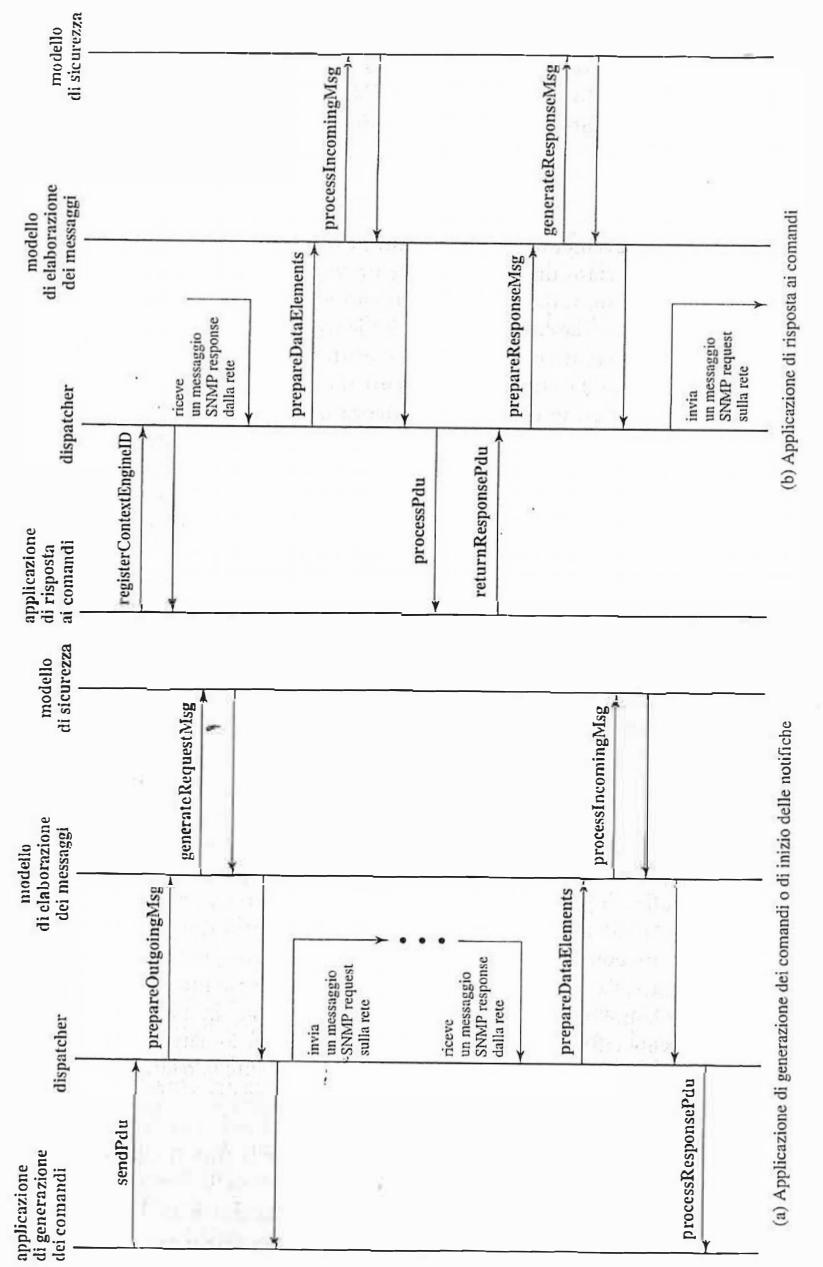


Figura 8.8 Flusso SNMPv3

Il documento RFC 2573 definisce, in termini generali, le procedure che ogni tipologia di applicazione segue nel generare le PDU da trasmettere e nell'elaborare le PDU ricevute. In tutti i casi considerati, le procedure sono definite in termini di interazione con il dispatcher, tramite le primitive ad esso associate.

Un'applicazione di generazione dei comandi utilizza le primitive sendPdu e processResponsePdu del dispatcher. La primitiva sendPdu fornisce informazioni al dispatcher circa la destinazione ultima, i parametri di sicurezza e la PDU effettiva che deve essere inviata. Il dispatcher invoca quindi, per preparare il messaggio, il modello di elaborazione di messaggi, che a sua volta richiama il modello di sicurezza. Il dispatcher inoltra il messaggio che è stato preparato a livello trasporto (per esempio, UDP) per la trasmissione. Se la preparazione del messaggio non va a buon fine, il valore restituito dalla primitiva sendPdu, generato dal dispatcher, rappresenta un'indicazione di errore. Se la preparazione del messaggio va a buon fine, il dispatcher assegna un identificatore sendPduHandle alla PDU in questione e restituisce il valore di tale identificatore al generatore di comandi. Il generatore di comandi memorizza il sendPduHandle in modo che possa far corrispondere la successiva PDU Response alla richiesta originale.

Il dispatcher trasmette ogni PDU Response che riceve alla corrispondente applicazione di generazione dei comandi, tramite la primitiva processResponsePdu.

Un'applicazione di risposta ai comandi utilizza quattro primitive del dispatcher (registerContextEngineID, unregisterContextEngineID, processPdu e returnResponsePdu) e una primitiva del sottosistema di controllo dell'accesso (isAccessAllowed).

La primitiva registerContextEngineID consente a un'applicazione di risposta ai comandi di mettersi in relazione con un motore SNMP al fine di elaborare alcune tipologie di PDU relative a un certo motore di contesto. Dopo che un'applicazione di risposta ai comandi si è registrata, tutti i messaggi asincroni ricevuti, contenenti la combinazione registrata di contextEngineID e pduType supportati, sono inviati a tale applicazione. Un'applicazione di risposta ai comandi può interrompere il legame con un motore SNMP utilizzando la primitiva unregisterContextEngineID.

Il dispatcher trasmette ciascuna PDU Request ricevuta contenente una richiesta alla corrispondente applicazione di risposta ai messaggi, tramite la primitiva processPdu. L'applicazione effettua a questo punto i seguenti passi.

- Esamina il contenuto della PDU Request. Il tipo di operazione deve corrispondere a uno dei tipi precedentemente registrati dall'applicazione in questione.
- Stabilisce se l'accesso necessario per effettuare l'operazione di gestione richiesta nella PDU considerata è consentito. A tale scopo, viene invocata la primitiva isAccessAllowed. Il parametro securityModel determina il modello di sicurezza che il sottosistema di controllo dell'accesso deve utilizzare per rispondere alla chiamata. Quest'ultimo determina se il principal che ha inviato la richiesta (securityName) al livello di sicurezza considerato (securityLevel) ha il permesso di richiedere l'operazione di gestione (viewType) sull'oggetto considerato (variableName) nel particolare contesto (contextName).
- Se l'accesso è consentito, l'applicazione di risposta ai comandi effettua l'operazione di gestione richiesta e prepara una PDU Response. Se l'accesso non viene consentito, l'applicazione prepara la PDU Response appropriata per segnalare tale situazione.

- Per inviare la PDU Response, l'applicazione di risposta ai comandi richiama il dispatcher tramite la primitiva returnResponsePdu.

Un'applicazione di generazione delle notifiche segue, in linea generale, le stesse procedure seguite da un'applicazione di generazione dei comandi. Nel caso si debba inviare una PDU InformRequest, vengono utilizzate sia la primitiva sendPdu che la primitiva processResponsePdu, con le stesse modalità con cui vengono utilizzate dalle applicazioni di generazione dei comandi. Nel caso in cui si debba inviare una PDU trap, viene utilizzata solo la primitiva sendPdu.

Un'applicazione di ricezione delle notifiche segue un sottoinsieme delle procedure di carattere generale seguite da un'applicazione di risposta ai comandi. L'applicazione di ricezione delle notifiche deve prima di tutto registrarsi per ricevere una PDU Inform e/o una PDU trap. Entrambi i tipi di PDU sono ricevuti tramite una primitiva processPdu. Nel caso di PDU Inform, per rispondere viene utilizzata una primitiva returnResponsePdu.

Un'applicazione proxy per l'inoltro utilizza le primitive del dispatcher per inoltrare messaggi SNMP. Tale applicazione gestisce quattro tipologie base di messaggi.

- Messaggi contenenti tipologie di PDU provenienti da un'applicazione di generazione dei comandi. In questo caso, l'applicazione proxy per l'inoltro di messaggi determina il motore SNMP di riferimento oppure un motore SNMP ad esso vicino, e invia la PDU Request appropriata.
- Messaggi contenenti tipologie di PDU generate da un'applicazione di generazione delle notifiche. L'applicazione per l'inoltro dei messaggi determina quali motori SNMP devono ricevere la notifica e invia una o più appropriate PDU Notification.
- Messaggi contenenti una PDU Response. L'applicazione proxy per l'inoltro dei messaggi determina quale tra le richieste o notifiche precedentemente inoltrate (se ne esistono) corrisponde alla PDU in questione, e invia un'appropriata PDU Response.
- Messaggi contenenti segnalazioni di report. Le PDU Report rappresentano comunicazioni tra motori SNMPv3. L'applicazione proxy per l'inoltro dei messaggi determina quale tra le richieste o notifiche precedentemente inoltrate (se ne esistono) corrisponde alla segnalazione riportata, e inoltra quest'ultima a chi ha originato la richiesta o la notifica.

Elaborazione di messaggi e modello di sicurezza utente

L'elaborazione dei messaggi si basa su un modello di tipo general-purpose e su uno specifico modello di sicurezza; la relazione tra i due modelli è illustrata nella Figura 8.8.

Modello di elaborazione dei messaggi

Il documento RFC 2572 definisce un modello di elaborazione dei messaggi di uso generale. Il compito di questo modello è quello di accettare PDU dal dispatcher, di incapsularle all'interno di messaggi e di invocare USM per inserire parametri di sicurezza all'interno dell'intestazione dei messaggi. Il modello di elaborazione dei messaggi accetta anche messaggi in entrata. In questo caso, il modello chiama USM, per elaborare i parametri di sicurezza contenuti nell'intestazione del messaggio, e successivamente invia la PDU contenuta nel messaggio al dispatcher.

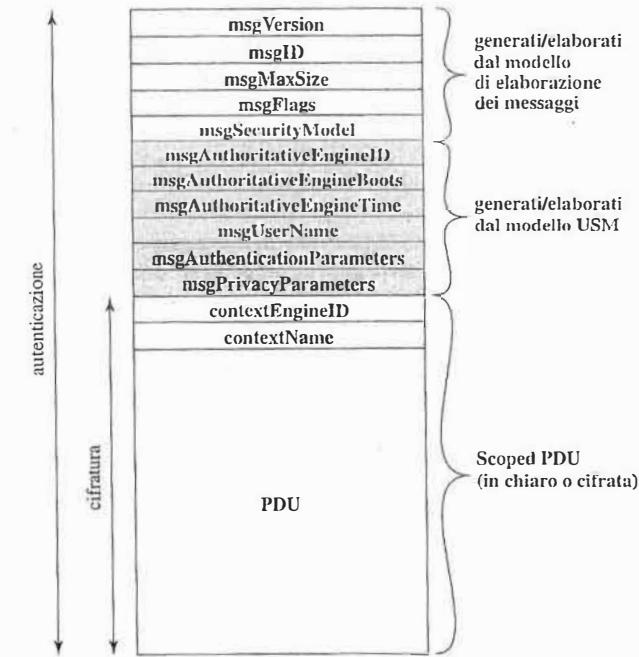


Figura 8.9 Formato del messaggio SNMPv3 con USM.

La Figura 8.9 illustra la struttura dei messaggi. Nel caso di messaggi in uscita, i primi cinque campi sono generati dal modello di elaborazione dei messaggi mentre, nel caso di messaggi in entrata, sono elaborati da tale modello. I sei campi successivi rappresentano parametri di sicurezza utilizzati da USM. Infine, l'unione della PDU, del contextEngineID e del contextName forma quella che viene chiamata **scoped PDU**, utilizzata per l'elaborazione della PDU. I primi cinque campi sono i seguenti.

- **msgVersion:** assume come valore snmpv3(3).
- **msgID:** è un identificatore univoco utilizzato tra due entità SNMP per coordinare i messaggi di richiesta e di risposta, e impiegato dall'applicazione che elabora il messaggio per coordinare l'elaborazione del messaggio tra i vari modelli utilizzati dai sottosistemi all'interno dell'architettura. Questo ID può assumere valori nell'intervallo da 0 a $2^{31} - 1$.
- **msgMaxSize:** stabilisce la dimensione massima dei messaggi che il mittente può gestire, espressa in byte. Tale dimensione può assumere valori nell'intervallo da 484 a $2^{31} - 1$. Tale campo rappresenta la dimensione massima di un segmento che il mittente può accettare da un altro motore SNMP (sia questo una risposta o un altro tipo di messaggio).

- **msgFlags:** stringa di byte contenente tre flag nei suoi tre bit meno significativi: reportableFlag, privFlag, authFlag. Se reportableFlag assume valore 1, allora si deve restituire una PDU Report al mittente in tutti quei casi che possono portare alla generazione di una PDU Report; quando il flag assume il valore 0, non deve essere inviata alcuna PDU Report. Il reportableFlag è posto al valore 1 dal mittente in tutti quei messaggi che contengono una PDU Request (Get, Set) o una PDU Inform, mentre è posto a 0 nel caso di messaggi che contengono PDU Response, Trap o Report. Il reportableFlag costituisce un ulteriore aiuto nel determinare quando inviare un Report ed è utilizzato solo nei casi in cui la porzione di messaggio relativa alla PDU non può essere decodificata (per esempio, quando il processo di decifratura fallisce a causa di una chiave non corretta). privFlag e authFlag sono inizializzati dal mittente per fornire indicazioni sul livello di sicurezza che è stato applicato al messaggio. Se privFlag assume il valore 1, significa che è stata utilizzata la cifratura, se assume il valore 0, significa che è stata applicata l'autenticazione. Sono possibili tutte le combinazioni ad eccezione di quella in cui privFlag assume il valore 1 e authFlag assume il valore 0, corrispondente all'applicazione della cifratura senza autenticazione.
- **msgSecurityModel:** identificatore che assume valori nell'intervallo da 0 a $2^{31} - 1$ e specifica quale modello di sicurezza è stato utilizzato dal mittente per preparare il messaggio. Il valore di questo campo indica quindi anche il modello di sicurezza che deve essere utilizzato dal ricevente per elaborare il messaggio. Valori riservati per tale campo includono il valore 1, nel caso di SNMPv1, 2 per SNMPv2c e 3 per SNMPv3.

Modello di sicurezza utente

Il documento RFC 2574 definisce il modello USM. USM fornisce servizi di autenticazione e riservatezza per SNMP, più precisamente, è progettato per fornire protezione contro le seguenti minacce.

- **Modifiche delle informazioni:** un'entità potrebbe alterare il contenuto di un messaggio in transito, generato da un'entità con la necessaria autorizzazione, in modo da causare operazioni di gestione non autorizzate, incluse le operazioni di modifica dei valori degli oggetti. Il punto focale di tale tipologia di attacco è che un'entità non autorizzata potrebbe cambiare un qualsiasi parametro di gestione, inclusi i parametri relativi alla configurazione, alle operazioni e all'accounting.
- **Attacco masquerade:** un'entità non autorizzata a compiere operazioni di gestione può tentare di effettuarle impersonando un'entità autorizzata a compierle.
- **Alterazione del flusso dei messaggi:** SNMP è stato progettato per operare al di sopra di un protocollo a livello trasporto privo di connessione. Esiste quindi il pericolo che i messaggi SNMP possano essere riordinati, ritardati o utilizzati per un attacco di replay (cioè duplicati) per eseguire operazioni di gestione non autorizzate. Ad esempio, un messaggio che ordina di effettuare il reboot di un certo dispositivo potrebbe essere copiato e utilizzato successivamente per un attacco di replay.
- **Divulgazione:** un'entità potrebbe osservare gli scambi che avvengono tra un gestore e un agente e venire così a conoscenza dei valori degli oggetti gestiti e di eventi notifi-

cibili. Ad esempio, l'osservazione di un comando set per la modifica delle password potrebbe consentire a un avversario di scoprire le nuove password.

USM non è stato concepito per garantire sicurezza contro i seguenti attacchi.

- **Negazione di servizio:** un avversario può impedire gli scambi tra un gestore e un agente.
- **Analisi del traffico:** un avversario può esaminare la tipologia di traffico tra i gestori e gli agenti.

La mancanza di protezione contro gli attacchi derivanti dalla negazione di servizio ha due principali giustificazioni. In primo luogo, tale tipologia di attacco è in molti casi indistinguibile dalle tipologie di guasti sulla rete con cui ogni applicazione di gestione di rete deve fare i conti; in secondo luogo, un attacco che comporta una negazione di servizio molto probabilmente renderà anche impossibile qualsiasi tipo di scambio. Questa tipologia di attacco dovrebbe quindi essere materia per una funzionalità di sicurezza a livello globale e non per una funzionalità contenuta all'interno di un protocollo di gestione di rete. Per quanto riguarda l'analisi del traffico, spesso le tipologie di traffico connesse alla gestione di rete sono prevedibili (per esempio, le entità possono essere gestite per mezzo di comandi SNMP inviati su base regolare da una o da poche stazioni di gestione); di conseguenza, non si ottiene alcun vantaggio significativo proteggendosi contro l'osservazione di tali tipologie di traffico.

Funzioni crittografiche

Le funzioni crittografiche definite per USM sono due: autenticazione e cifratura. Per fornire supporto a queste due funzioni, un motore SNMP necessita di due valori: una chiave per assicurare riservatezza (privKey) e una chiave di autenticazione (authKey). Per le seguenti tipologie di utenti vengono mantenuti valori distinti per le due chiavi.

- **Utenti locali:** un qualsiasi principal presso il motore SNMP considerato per cui sono autorizzate operazioni di gestione.
- **Utenti remoti:** un qualsiasi principal presso un motore SNMP remoto con cui effettuare comunicazioni.

Tali valori sono attribuiti utente, mantenuti per ciascun utente di interesse. I valori di privKey e authKey non sono accessibili tramite SNMP.

USM consente l'utilizzo di uno tra due protocolli di autenticazione alternativi: HMAC-MD5-96 e HMAC-SHA-96. Il protocollo HMAC, descritto nel Capitolo 3, utilizza una funzione hash sicura e una chiave segreta per generare un MAC. Nel caso di HMAC-MD5-96, viene utilizzato HMAC con MD5 come funzione hash. Come ingresso dell'algoritmo HMAC viene utilizzata un'authKey di 16 byte (128 bit). L'algoritmo produce in uscita 128 bit che sono troncati a 12 byte (96 bit). Nel caso di HMAC-SHA-96, la funzione hash utilizzata è SHA-1. L'authKey ha una lunghezza pari a 20 byte. L'algoritmo produce in uscita 20 byte che, ancora una volta, sono troncati a 12 byte.

Per la cifratura, USM utilizza la modalità CBC di DES. Il protocollo di cifratura riceve in ingresso una privKey di lunghezza pari a 16 byte. I primi 8 (64 bit) della privKey sono utilizzati come chiave DES. Dato che DES richiede chiavi di lunghezza di 56 bit, non

viene preso in considerazione il bit meno significativo di ciascun byte. Per la modalità CBC, è necessario un vettore di inizializzazione di 64 bit. Per generare tale vettore, si utilizza il valore contenuto negli ultimi 8 byte della privKey.

Motori con o senza autorità

Tutte le volte che si trasmette un messaggio, una delle due entità coinvolte – il mittente o il ricevente – gioca il ruolo di motore SNMP con autorità (*authoritative engine*). La scelta di quale dei due entità in comunicazione deve giocare questo ruolo si basa sulle seguenti regole.

- Quando un messaggio SNMP contiene un payload che necessita di una risposta (ad esempio: una PDU Get, GetNext, GetBulk, Set o Inform), viene designato il ricevente del messaggio come entità con autorità.
- Quando il messaggio SNMP contiene un payload che non presuppone una risposta (ad esempio: una PDU Trap SNMPv2, Response o Report), allora è il mittente del messaggio che ricopre il ruolo di entità con autorità.

Ciò implica che il ricevente diventa l'entità con autorità nel caso di messaggi inviati per conto di un'applicazione di generazione dei comandi o nel caso di messaggi Inform inviati da un'applicazione di generazione delle notifiche. Per contro, nel caso di messaggi inviati per conto di un'applicazione di risposta ai messaggi o nel caso di messaggi Trap inviati da un'applicazione di generazione delle notifiche, l'entità con autorità è il mittente. Questo schema ha un duplice scopo.

- La tempestività di un messaggio è determinata in base a un clock che viene gestito dal motore con autorità. Quando un motore con autorità invia un messaggio (Trap, Response, Report), il messaggio contiene il valore attuale del proprio clock, in modo che i riceventi senza autorità possano sincronizzarsi rispetto a questo. Nel caso in cui il messaggio sia inviato da un motore senza autorità (Get, GetNext, GetBulk, Set, Inform), il messaggio contiene la stima attuale del valore temporale alla destinazione, in modo che la destinazione possa valutare la tempestività del messaggio.
- Un processo di localizzazione delle chiavi, descritto più avanti in questo capitolo, consente a un unico principal di possedere chiavi che sono memorizzate in svariati motori; queste chiavi sono poste presso il motore con autorità in modo che il principal sia responsabile di un'unica chiave ma nel contempo si evitino i rischi di sicurezza derivanti dal memorizzare diverse copie della stessa chiave su una rete distribuita.

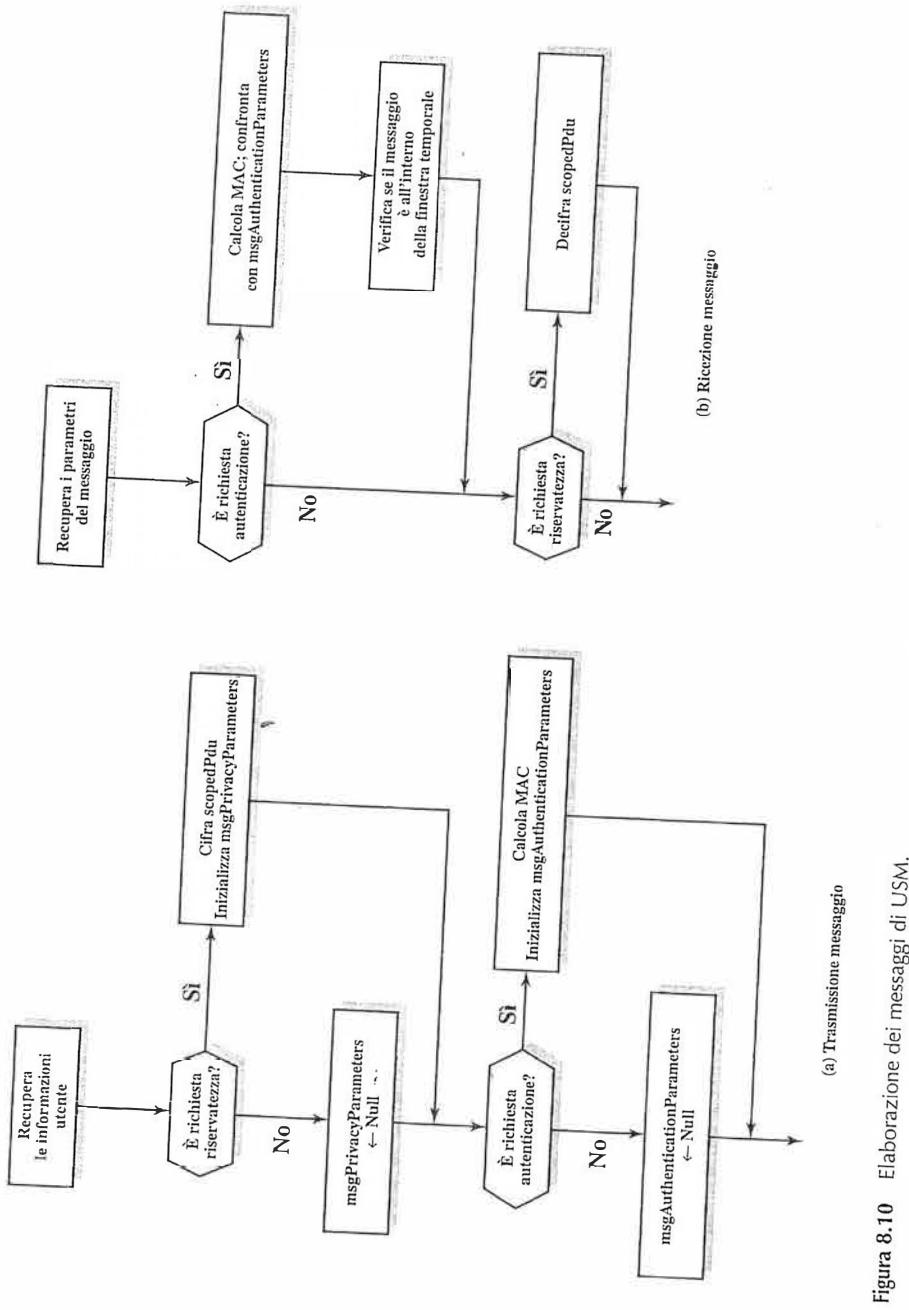
La scelta di designare colui che riceve PDU generate da un'applicazione di generazione dei comandi o PDU Inform come motore con autorità è sicuramente ragionevole. Tale motore ha quindi anche la responsabilità di verificare la tempestività dei messaggi. Con questo schema il ritardo di una risposta, una trap o il loro utilizzo per un attacco di replay, provoca pochi danni. Comunque, le PDU relative a un'applicazione di generazione dei comandi e, in una qualche misura, le PDU Inform comportano operazioni di gestione, quali operazioni di lettura o di modifica degli oggetti MIB. È dunque importante garantire che queste PDU non siano ritardate o utilizzate per un attacco di replay, in quanto ciò potrebbe causare effetti indesiderati.

Parametri dei messaggi USM

Quando un messaggio in uscita è inoltrato dal sottosistema di elaborazione dei messaggi all'USM, questi inizializza i parametri relativi alla sicurezza contenuti nell'intestazione del messaggio. Al contrario, quando un messaggio in arrivo è trasmesso all'USM da parte del sottosistema di elaborazione dei messaggi, l'USM elabora i valori contenuti nei campi sopra citati. I parametri relativi alla sicurezza sono i seguenti.

- **msgAuthoritativeEngineID:** valore di snmpEngineID del motore SNMP con autorità che è coinvolto nello scambio del messaggio in questione. Quindi, tale valore si riferisce alla sorgente nel caso di Trap, Response o Report, mentre si riferisce alla destinazione nel caso di Get, GetNext, GetBulk, Set o Inform.
- **msgAuthoritativeEngineBoots:** valore di snmpEngineBoots del motore SNMP con autorità coinvolto nello scambio del messaggio in questione. snmpEngineBoots è un intero compreso nell'intervallo che va da 0 a $2^{31} - 1$. Tale valore intero rappresenta il numero di volte, dalla sua configurazione iniziale, che il motore SNMP in questione ha effettuato un'operazione di inizializzazione o reinizializzazione.
- **msgAuthoritativeEngineTime:** valore di snmpEngineTime per il motore SNMP con autorità coinvolto nello scambio del messaggio in questione. snmpEngineTime è un intero compreso nell'intervallo da 0 a $2^{31} - 1$, che denota il numero di secondi trascorsi dal momento in cui il motore SNMP con autorità ha incrementato per l'ultima volta snmpEngineBoots. Ogni motore SNMP con autorità è responsabile dell'incremento del proprio valore di snmpEngineTime, incremento che deve avvenire una volta al secondo. Un motore senza autorità è invece responsabile dell'incremento della sua versione di snmpEngineTime per ciascun motore con autorità remoto con cui comunica.
- **msgUserName:** utente (principal) per conto del quale avviene lo scambio del messaggio.
- **msgAuthenticationParameters:** parametro che assume un valore nullo se non viene utilizzata l'autenticazione per lo scambio in questione; altrimenti è un parametro di autenticazione. Nella definizione corrente di USM, il parametro di autenticazione è un MAC generato tramite HMAC.
- **msgPrivacyParameters:** parametro che assume un valore nullo se non viene utilizzato un meccanismo per garantire la riservatezza durante lo scambio in questione. In caso contrario, è un parametro di riservatezza. Nella definizione attuale di USM, il parametro di riservatezza è un valore utilizzato per generare l'IV dell'algoritmo DES in modalità CBC.

La Figura 8.10 riassume il funzionamento di USM. Quando è trasmesso un messaggio, viene prima effettuata la cifratura, se necessario. La scoped PDU viene cifrata e posta nel payload del messaggio, mentre il valore msgPrivacyParameters è inizializzato al valore necessario per generare IV. Successivamente, si effettua l'autenticazione, se necessario. L'intero messaggio, inclusa la scoped PDU, è dato in ingresso a HMAC e il risultante codice di autenticazione è posto nel parametro msgAuthenticationParameters. Per i messaggi in arrivo, viene prima effettuata l'autenticazione, se necessaria. USM verifica per prima cosa il MAC nel messaggio in arrivo rispetto al MAC calcolato direttamente; se i due valori coincidono,



il messaggio è considerato autentico (cioè proviene dalla sorgente presunta e non è stato alterato durante la trasmissione). A questo punto USM verifica se il messaggio è all'interno della finestra temporale consentita, secondo le modalità che verranno di seguito illustrate. Se il messaggio non ricade in tale finestra, viene scartato perché considerato non autentico. Infine, se la scoped PDU era stata cifrata, USM effettua la decifratura e restituisce il testo in chiaro.

Meccanismi USM per verificare la tempestività dei messaggi

USM prevede un insieme di meccanismi per verificare la tempestività dei messaggi, che possono essere utilizzati come protezione contro il ritardo dei messaggi e gli attacchi di replay. Ogni motore SNMP che potrebbe nel corso del tempo assumere il ruolo di motore con autorità deve mantenere due oggetti, snmpEngineBoots e snmpEngineTime, che si riferiscono al tempo locale. Quando un motore SNMP è installato per la prima volta, i valori di questi due oggetti sono posti a 0. Successivamente, snmpEngineTime viene incrementato una volta al secondo. Ogni volta che snmpEngineTime raggiunge il suo valore massimo ($2^{31} - 1$), snmpEngineBoots viene incrementato come se fosse stato effettuato un reboot del sistema, mentre snmpEngineTime è posto a 0 e comincia ad essere incrementato nuovamente. Utilizzando un meccanismo di sincronizzazione, un motore senza autorità mantiene una sua stima personale del valore del clock di ogni motore con autorità con cui comunica. Queste stime sono poste in ogni messaggio in uscita e consentono al motore con autorità che riceve il messaggio di determinare se il messaggio in arrivo non ha subito ritardi.

Il meccanismo di sincronizzazione opera con le seguenti modalità. Un motore senza autorità mantiene, per ogni motore con autorità a lui noto, una copia locale di tre variabili.

- **snmpEngineBoots**: valore più recente assunto da snmpEngineBoots per il motore con autorità remoto.
- **snmpEngineTime**: valore stimato dal motore senza autorità per snmpEngineTime del motore con autorità remoto. Tale valore è sincronizzato rispetto al motore con autorità remoto tramite un processo che verrà descritto in seguito. Nell'intervallo che intercorre tra due eventi di sincronizzazione, questo valore è logicamente incrementato una volta al secondo al fine di mantenere una forma di sincronizzazione lasca con il motore con autorità remoto.
- **latestReceivedEngineTime**: il più alto valore di msgAuthoritativeEngineTime inviato al motore in questione dal motore con autorità remoto; tale valore è aggiornato tutte le volte che viene ricevuto un valore più elevato di msgAuthoritativeEngineTime. Lo scopo di questa variabile è fornire protezione contro attacchi di replay che impedirebbero l'incremento del valore previsto per snmpEngineTime dal motore senza autorità.

Il motore mantiene queste tre variabili per ogni motore con autorità a lui noto. Logicamente, i valori di queste variabili sono memorizzati in qualche tipologia di memoria cache, indirizzati tramite il valore di snmpEngineID, che identifica in modo univoco ogni motore con autorità remoto.

Per fare in modo che i motori senza autorità possano mantenere la sincronizzazione temporale, ciascun motore con autorità inserisce – in ogni messaggio Response, Report o Trap in uscita – i valori correnti per il tempo di boot e il tempo corrente, oltre al valore di

snmpEngine-ID. Tali valori sono inseriti rispettivamente nei campi msgAuthoritativeEngineBoots, msgAuthoritativeEngineTime e msgAuthoritativeEngineID. Se il messaggio è autentico ed è all'interno della finestra temporale consentita, il motore senza autorità che riceve il messaggio aggiorna le proprie variabili locali (snmpEngineBoots, snmpEngineTime e latestReceivedEngineTime) relative al motore con autorità remoto che ha inviato il messaggio, in base alle seguenti regole.

1. Viene effettuato un aggiornamento se almeno una delle condizioni seguenti è verificata:

- $(\text{msgAuthoritativeEngineBoots} > \text{snmpEngineBoots}) \text{ OR}$
- $[(\text{msgAuthoritativeEngineBoots} = \text{snmpEngineBoots}) \text{ AND}$
 $(\text{msgAuthoritativeEngineTime} > \text{latestReceivedEngineTime})]$

La prima condizione asserisce che dovrebbe avvenire un aggiornamento se il valore del tempo di boot del motore con autorità è stato incrementato dal momento in cui è avvenuto l'ultimo aggiornamento. La seconda condizione stabilisce che, nel caso in cui il valore del tempo di boot non sia stato incrementato, debba avvenire un aggiornamento se il valore del tempo corrente per il motore che ha ricevuto il messaggio è maggiore di quello ricevuto con l'ultimo messaggio. Tale condizione non è verificata se due messaggi non arrivano nell'ordine corretto, situazione che può verificarsi nel caso in cui sia stato effettuato un attacco di replay; in entrambi i casi, il motore che riceve il messaggio non effettua alcun aggiornamento.

2. Nel caso in cui sia richiesta un'operazione di aggiornamento, vengono effettuate le seguenti modifiche:

- snmpEngineBoots è posto uguale al valore di msgAuthoritativeEngineBoots
- snmpEngineTime è posto uguale al valore di msgAuthoritativeEngineTime
- latestReceivedEngineTime è posto uguale al valore di msgAuthoritativeEngineTime

Considerando la questione da un altro punto di vista, si può osservare come, se $\text{msgAuthoritativeEngineBoots} < \text{snmpEngineBoots}$, allora non viene effettuata alcuna operazione di aggiornamento. In questo caso, il messaggio non viene considerato autentico e non deve quindi essere preso in considerazione. Non viene effettuata alcuna operazione di modifica anche nel caso in cui $\text{msgAuthoritativeEngineBoots} = \text{snmpEngineBoots}$ ma $\text{msgAuthoritativeEngineTime} < \text{latestReceivedEngineTime}$. In questo caso, è possibile che il messaggio sia autentico, ma l'ordine può non essere quello corretto, nel qual caso una modifica di snmpEngineTime non è giustificata.

Si noti come il meccanismo di sincronizzazione venga utilizzato solo nel caso in cui sia in uso il servizio di autenticazione per il messaggio in questione e l'autenticità del messaggio sia stata provata tramite HMAC. Tale vincolo è fondamentale in quanto l'autenticazione comprende msgAuthoritativeEngineID, msgAuthoritativeEngineBoots e msgAuthoritativeEngineTime, assicurando quindi la validità dei loro valori.

SNMPv3 impone che un messaggio debba essere ricevuto entro una ragionevole finestra temporale, al fine di evitare ritardi ed attacchi di replay. La finestra temporale dovrebbe essere scelta in modo da risultare la più piccola possibile, data la precisione dei clock coinvolti, i ritardi di andata e ritorno nella comunicazione e la frequenza con cui i clock sono sincronizzati. Nel caso in cui si scelga una finestra troppo piccola, può succedere che messaggi autentici ven-

gano rifiutati perché considerati non autentici. Per contro, una finestra troppo grande aumenta la vulnerabilità a ritardi intenzionali dei messaggi associati ad attacchi.

In seguito verrà considerato il caso più rilevante, cioè quello in cui colui che riceve il messaggio è un motore con autorità; i controlli di tempestività nel caso di motore senza autorità differiscono in modo lieve. Per ogni messaggio in arrivo che è stato autenticato e con il valore di msgAuthoritativeEngineID uguale al valore di snmpEngineID per il motore che lo riceve, il motore confronta i valori di msgAuthoritativeEngineBoots e msgAuthoritativeEngineTime contenuti nel messaggio in arrivo con i propri valori di snmpEngineBoots e snmpEngineTime. Il messaggio in arrivo è considerato fuori dai limiti della finestra temporale se si verifica almeno una delle seguenti condizioni:

- $\text{snmpEngineBoots} = 2^{31} - 1 \text{ OR}$
- $\text{msgAuthoritativeEngineBoots} \neq \text{snmpEngineBoots} \text{ OR}$
- il valore msgAuthoritativeEngineTime differisce dal valore di snmpEngineTime per più di ± 150 secondi.

La prima condizione stabilisce che se snmpEngineBoots assume il suo valore massimo, nessun messaggio in arrivo può essere considerato autentico. La seconda condizione impone che un messaggio debba avere un tempo di boot uguale al tempo locale di boot del motore in questione; ad esempio, se il motore locale ha effettuato un reboot e, da quel momento, il motore remoto non si è più sincronizzato con quello locale, i messaggi che provengono dal motore remoto sono considerati non autentici. L'ultima condizione stabilisce che il tempo corrente contenuto nel messaggio in arrivo deve essere maggiore del tempo locale meno 150 secondi e minore del tempo locale più 150 secondi.

Se un messaggio è considerato al di fuori della finestra temporale, allora è giudicato non autentico. In questo caso, viene restituita un'indicazione di errore (notInTimeWindow) al modulo chiamante.

Come nel caso della sincronizzazione, la verifica della tempestività del messaggio è effettuata solo quando viene utilizzato il servizio di autenticazione e il messaggio è autentico, caratteristica che assicura la validità dei campi contenuti nell'intestazione del messaggio.

Localizzazione delle chiavi

Uno dei requisiti per utilizzare i servizi di autenticazione e riservatezza di SNMPv3 è che, per ogni comunicazione tra un principal su un motore senza autorità e un motore con autorità remoto, esista una chiave segreta di autenticazione e una chiave segreta per assicurare la riservatezza, condivise dalle due parti in comunicazione. Queste chiavi consentono a un utente collegato a un motore senza autorità (in genere un sistema di gestione) di far uso dei servizi di autenticazione e riservatezza per le comunicazioni con i sistemi remoti con autorità che l'utente gestisce (in genere sistemi agente). Il documento RFC 2574 fornisce le linee guida per la creazione, l'aggiornamento e la gestione di queste chiavi.

Per alleggerire il carico che la gestione delle chiavi comporta presso i principal, a ciascuno di questi è richiesto il mantenimento di un'unica chiave di autenticazione e di un'unica chiave di cifratura. Tali chiavi non sono memorizzate nella MIB e non sono accessibili tramite SNMP. In questo paragrafo, verranno esaminate in primo luogo le tecniche per generare queste chiavi partendo da una password. Successivamente, si considererà il concetto

di localizzazione delle chiavi, che consente a un principal di condividere un'unica chiave di autenticazione e un'unica chiave di cifratura con ogni motore remoto, mantenendo però presso di sé un'unica chiave di autenticazione e un'unica chiave di cifratura. Le tecniche che verranno illustrate sono state proposte per la prima volta in [BLUM97a].

Un utente ha bisogno di una chiave di cifratura di 16 byte e di una chiave di autenticazione di 16 o 20 byte. Nel caso in cui i possessori delle chiavi siano persone fisiche, è auspicabile che l'utente possa utilizzare come chiave una password a lui comprensibile, piuttosto che una stringa di bit. A tal scopo, il documento RFC 2574 specifica un algoritmo che traduce una password utente in una chiave di 16 o 20 byte. USM non impone alcuna restrizione sulla password, ma le politiche di gestione a livello locale dovrebbero imporre all'utente di non utilizzare password che possano essere facilmente scoperte.

La generazione della chiave a partire dalla password è effettuata secondo le seguenti modalità.

- La password utente viene presa come ingresso e viene restituita una stringa di lunghezza pari a 2^{20} byte (1.048.576 byte) ottenuta ripetendo il valore della password il numero di volte necessario ed eventualmente troncando l'ultima ripetizione, per formare la stringa digest0. Ad esempio, nel caso di una password formata da 8 caratteri (2^3 byte), la password verrebbe concatenata ripetutamente a sé stessa per un numero di volte pari a 2^{17} , per formare la stringa digest0.
- Nel caso sia richiesta una chiave di 16 byte, si genera la stringa digest1, applicando MD5 a digest0. Se invece è richiesta una chiave di 20 byte, digest1 è ottenuto applicando l'algoritmo SHA-1 a digest0. Il risultato è la chiave utente.

Uno dei vantaggi di questa tecnica è che rende molto più lento un attacco a forza bruta basato su dizionario, attacco in cui un avversario prova un numero considerevole di password potenziali, generando da ognuna la corrispondente chiave e verificando poi se la chiave ottenuta funziona con i dati di autenticazione o di cifratura in suo possesso. Ad esempio, se un avversario intercetta un messaggio autenticato, potrebbe provare a generare il valore HMAC mediante molteplici, possibili chiavi utente. Se i tentativi vanno a buon fine, l'avversario può ipotizzare di aver scoperto la password. Il processo appena descritto, aumenta notevolmente il tempo necessario per perpetrare questo attacco.

Un ulteriore vantaggio è che questa tecnica svincola le chiavi utente da un particolare **sistema di gestione di rete** (NMS, *network management system*). Nessun NMS ha necessità di memorizzare i valori delle chiavi utente. Al contrario, quando è necessario, la chiave utente viene generata sulla base della corrispondente password. In [BLUM97b] sono elencate le seguenti motivazioni per giustificare l'utilizzo di un approccio basato su password, indipendente dallo specifico NMS.

- Se si utilizzasse l'approccio di memorizzare le chiavi invece di generarle dalle password, un'alternativa sarebbe quella di predisporre un repository centralizzato in cui memorizzare le chiavi segrete. Ma questo tipo di soluzione influenza negativamente l'affidabilità generale e rende impossibile la risoluzione dei problemi nel caso in cui il repository non sia accessibile quando necessario.
- D'altro canto, mantenere più di una copia del repository metterebbe in pericolo la sicurezza globale in quanto l'avversario dispone di più bersagli da colpire.

- I repository devono essere mantenuti in posti sicuri, sia quando se ne mantenga uno solo centralizzato sia quando ne esistano più copie. Questo tipo di precauzione può ridurre la possibilità di stabilire punti di ripristino durante situazioni di emergenza, cioè diventa più difficile la risoluzione dei problemi quando dei segmenti della rete non sono operativi in modo non prevedibile o sono inaccessibili per un periodo di tempo non stimabile.

Si potrebbe anche utilizzare un'unica password per generare un'unica chiave, da utilizzare tanto per l'autenticazione che per la cifratura. Uno schema più sicuro è utilizzare due password, una per generare una chiave di autenticazione e l'altra per generare una chiave di cifratura diversa.

Nel documento RFC 2574 è introdotta la nozione di chiave localized (localizzata). Una chiave localized è una chiave segreta condivisa tra un utente e un unico motore SNMP con autorità. L'obiettivo è fare in modo che l'utente debba mantenere un'unica chiave (o due chiavi nel caso siano richieste sia l'autenticazione che la riservatezza) e di conseguenza debba ricordarsi solo una password (o due). Ciascun utente condivide un'informazione segreta diversa con ogni motore SNMP con autorità. Si indica con il termine **localizzazione delle chiavi** il processo attraverso cui una singola chiave utente è trasformata in un insieme di chiavi uniche, una per ciascun motore SNMP remoto. [BLUM97a] fornisce una motivazione per questa strategia, riassunta in seguito. La gestione delle chiavi si prefigge i seguenti obiettivi.

- Ogni sistema agente SNMP in una rete distribuita possiede la propria chiave univoca per ciascun utente autorizzato alla sua gestione. Se esistono molti utenti autorizzati a compiti di gestione, l'agente avrà una singola chiave di autenticazione e una singola chiave di cifratura per ciascuno di questi utenti. Di conseguenza, la compromissione della chiave di un certo utente non influenza le chiavi di tutti gli altri.
- Ciascun utente possiede chiavi diverse per agenti diversi. Questo implica che la compromissione di un agente comporta la compromissione solo delle chiavi utente per quello specifico agente, mentre quelle relative ad altri agenti non sono influenzate.
- La gestione di rete può essere effettuata da ogni punto della rete, indipendentemente dalla disponibilità di un NMS preconfigurato. Tale caratteristica consente a un utente di effettuare operazioni di gestione da una qualsiasi stazione di gestione. Questa possibilità è fornita dall'algoritmo di trasformazione delle password in chiavi precedentemente descritto.

Ecco invece un elenco di aspetti negativi.

- Un utente deve ricordarsi (o comunque gestire) un numero elevato di chiavi e tale numero cresce con l'aggiunta di nuovi agenti da gestire.
- Un avversario che entra in possesso di una chiave per un certo agente è in grado di impersonare un qualsiasi altro agente per un qualsiasi utente, o un qualsiasi utente per un qualsiasi agente.

Per dare una risposta a tali problemi, una singola chiave utente è associata tramite una funzione unidirezionale non reversibile (per esempio, una funzione hash sicura) a varie chiavi localized, relative a diversi motori autenticati (agenti differenti). La procedura per attuare tale associazione è la seguente.

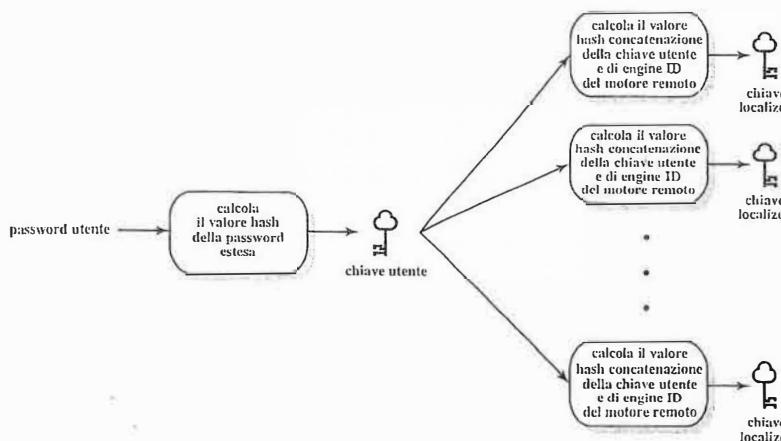


Figura 8.11 Localizzazione delle chiavi.

- Si genera la stringa digest2 concatenando la stringa digest1 (precedentemente descritta), il valore di snmpEngineID per il motore con autorità e la stringa digest1.
- Se è richiesta una chiave di 16 byte, si prende il valore hash generato da MD5 per la stringa digest2. Nel caso sia necessaria una chiave di 20 byte, si prende il valore hash generato da SHA-1 per la stringa digest2. Il risultato è la chiave utente localized.

La chiave localized risultante può a questo punto essere configurata presso il sistema dell'agente mediante una qualche modalità sicura. Dato che MD5 e SHA-1 sono unidirezionali, è impossibile per un avversario scoprire una chiave utente anche nel caso in cui l'avversario scoprissse una chiave localized. La Figura 8.11 riassume il processo di localizzazione delle chiavi.

Controllo dell'accesso basato su viste

Il controllo dell'accesso è una funzione di sicurezza effettuata a livello di PDU. Un documento di controllo dell'accesso definisce meccanismi per determinare quando una richiesta di accesso a un oggetto di una MIB locale, effettuata da un principal remoto, debba essere autorizzata. È indubbio che possono essere definiti molti meccanismi di controllo dell'accesso. I documenti SNMPv3 definiscono il modello di controllo dell'accesso basato su viste (VACM, *view-based access control*), che utilizza a sua volta una MIB che contiene la politica di controllo dell'accesso per il corrispondente agente e rende possibile l'utilizzo di una configurazione remota.

Elementi del modello VACM

Il documento RFC 2575 definisce i cinque elementi alla base di VACM: gruppi, livello di sicurezza, contesti, viste MIB e politica di accesso.

Un **gruppo** è definito come un insieme di zero o più tuple della forma <securityModel, securityName> tramite le quali si può accedere a un oggetto gestito con SNMP. La compo-

nente securityName si riferisce a un principal. Tutti i principal appartenenti allo stesso gruppo hanno identici diritti di accesso. A ogni gruppo è associato un unico groupName. Il concetto di gruppo è uno strumento utile per classificare i gestori rispetto ai diritti di accesso. Ad esempio, i gestori di livello più alto possono avere tutti lo stesso insieme di diritti di accesso, mentre i gestori di livello intermedio possono essere caratterizzati da un insieme diverso di diritti di accesso.

Ogni combinazione di valori per securityModel e securityName può appartenere al massimo a un gruppo. Di conseguenza, per un dato agente, un certo principal le cui comunicazioni sono protette da uno specifico modello di sicurezza (denotato dal valore di securityModel) può essere incluso in un solo gruppo.

I diritti di accesso di un gruppo possono differire a seconda del **livello di sicurezza** del messaggio che contiene la richiesta di accesso. Ad esempio, un agente può consentire operazioni di sola lettura nel caso di richieste contenute in un messaggio non autenticato, mentre può richiedere l'autenticazione nel caso di accessi in scrittura. Inoltre, nel caso di particolari oggetti contenenti informazioni sensibili, l'agente può richiedere che la comunicazione della richiesta e della risposta avvenga mediante il servizio che assicura la riservatezza.

Un **contesto MIB** è un sottoinsieme di istanze di oggetti contenute nella MIB locale, cui è stato associato un nome. I contesti sono utili per raggruppare oggetti in insiemi caratterizzati da differenti politiche di accesso.

Il concetto di contesto è connesso al controllo dell'accesso. Quando una stazione di gestione interagisce con un agente per accedere all'informazione di gestione memorizzata presso di lui, l'interazione avviene tra il principal incaricato della gestione e il motore SNMP dell'agente. I privilegi di accesso sono espressi tramite una vista MIB che è relativa al principal e al suo contesto. I contesti hanno le seguenti caratteristiche chiave.

- Un'entità SNMP, identificata in modo univoco da un valore di contextEngineID, può mantenere più di un contesto.
- Un oggetto o una sua istanza può apparire in più di un contesto.
- Nel caso di più contesti, al fine di identificare in modo univoco una certa istanza di un oggetto, è necessario identificare, oltre al tipo dell'oggetto e all'istanza, i suoi valori per contextName e per contextEngineID.

Spesso si desidera limitare gli accessi di un certo gruppo solo a un sottoinsieme degli oggetti gestiti presso un agente. A tal fine, l'accesso a un contesto avviene tramite una **vista MIB**, che denota un insieme specifico di oggetti gestiti (e optionalmente specifiche istanze degli oggetti). VACM utilizza un meccanismo potente e flessibile per specificare le viste MIB. Tale meccanismo è basato sul concetto di sottoalbero di viste e di famiglie di viste. Una vista MIB è definita in termini di un insieme, o famiglia, di sottoalberi, dove ciascun sottoalbero può essere contenuto o non contenuto nella vista stessa.

Gli oggetti contenuti in una base di dati locale sono organizzati in una struttura gerarchica, o albero, basata sugli identificatori degli oggetti. Questa base di dati locale contiene un sottoinsieme di tutti i tipi di oggetti definiti in base al documento Internet standard Structure of Management Information (SMI). La base di dati contiene le istanze di oggetti i cui identificatori sono conformi alle convenzioni SMI.

SNMPv3 prevede il concetto di sottoalbero. Un sottoalbero è semplicemente costituito da un nodo nella gerarchia di oggetti della MIB e da tutti i sottoelementi di tale

nodo. In modo più formale, un sottoalbero può essere definito come l'insieme di tutti gli oggetti e di tutte le loro istanze il cui nome è preceduto dal prefisso comune OBJECT IDENTIFIER ASN.1. Il più lungo prefisso comune in tutte le istanze nel sottoalbero è l'identificatore del nodo padre del sottoalbero in considerazione.

A ciascuna voce della tabella `vacmAccessTable` sono associate tre viste MIB, corrispondenti ai privilegi di lettura, scrittura e notifica. Ogni vista MIB consiste in un insieme di sottoalberi. Per ogni sottoalbero contenuto nella vista MIB deve essere specificato se è incluso o escluso dalla vista. In altri termini, la vista MIB conterrà o meno tutte le istanze degli oggetti presenti in quel sottoalbero. Inoltre, per diminuire la quantità di informazioni di configurazione necessarie nel caso in cui sia richiesto un controllo dell'accesso con un livello di granularità fine (per esempio, controllo dell'accesso a livello di istanza), viene definita una maschera di vista (*view mask*).

VACM consente di configurare un motore SNMP in modo da consentire un certo insieme di diritti di accesso, che costituiscono una **politica di accesso**. Il risultato di una richiesta di accesso dipende dai fattori qui di seguito elencati.

- Il **principal** che effettua la richiesta. Con VACM, un agente può consentire differenti accessi a utenti diversi. Ad esempio, un sistema di gestione responsabile della configurazione globale di rete, può avere ampi privilegi per modificare elementi all'interno della MIB locale, mentre un sistema di gestione di livello intermedio con responsabilità di monitoraggio può avere accessi in sola lettura e i suoi privilegi possono essere ulteriormente limitati, permettendo l'accesso solo a una parte della MIB locale. Come illustrato in precedenza, i principal vengono assegnati a specifici gruppi e le politiche di accesso vengono specificate a livello di gruppo.
- Il **livello di sicurezza** del messaggio SNMP contenente la richiesta di accesso. Generalmente, un agente richiede l'utilizzo dell'autenticazione per i messaggi contenenti richieste di operazioni set (operazione di scrittura).
- Il **modello di sicurezza** utilizzato per elaborare il messaggio che contiene la richiesta. Se un agente dispone di più modelli di sicurezza, questi può essere configurato per fornire livelli diversi di accesso a richieste inoltrate tramite messaggi elaborati da differenti modelli di sicurezza. Ad esempio, alcuni elementi nella MIB possono essere accessibili se il messaggio proviene tramite USM, e non accessibili se il modello di sicurezza utilizzato è SNMPv1.
- Il **contesto MIB** relativo alla richiesta.
- La specifica **istanza di oggetto** cui la richiesta di accesso si riferisce. Alcuni oggetti contengono informazioni più sensibili o critiche di altri, e quindi le politiche di accesso devono dipendere dalla specifica istanza richiesta.
- Il **tipo di accesso** richiesto (lettura, scrittura, notifica). Lettura, scrittura e notifica rappresentano tre distinte operazioni di gestione; ad esse si possono quindi applicare differenti politiche di controllo dell'accesso.

Controllo dell'accesso

Un'applicazione SNMP invoca il VACM tramite la primitiva `isAccessAllowed`, fornendo in ingresso a tale primitiva i parametri `securityModel`, `securityName`, `securityLevel`,

`viewType`, `contextName` e `variableName`. Tutti questi parametri sono necessari per decidere se una richiesta di accesso può essere accordata o meno. In altri termini, il sottosistema di controllo dell'accesso è definito in modo tale da fornire uno strumento molto flessibile per la configurazione del controllo dell'accesso presso l'agente, basato su sei variabili distinte.

La Figura 8.12, adattamento di una figura del documento RFC 2575, fornisce un'utile chiave di lettura per le variabili in ingresso al meccanismo di controllo dell'accesso; illustra inoltre il ruolo delle varie tabelle contenute nella MIB VACM nella valutazione di una richiesta di accesso.

- **who (chi)**: la coppia costituita dai valori di `securityModel` e `securityName` specifica il soggetto dell'operazione; identifica cioè un particolare principal le cui comunicazioni sono protette da un certo modello di sicurezza (indicato dal valore di `securityModel`). Tale coppia di valori appartiene al più a un gruppo per il motore SNMP in considerazione. La tabella `vacmSecurityToGroupTable` fornisce il valore di `groupName` corrispondente a una data coppia di valori per `securityModel` e `securityName`.
- **where (dove)**: il valore di `contextName` specifica come localizzare l'oggetto gestito cui la richiesta di accesso corrisponde. La tabella `vacmContextTable` contiene la lista dei valori di `contextNames` ammessi.

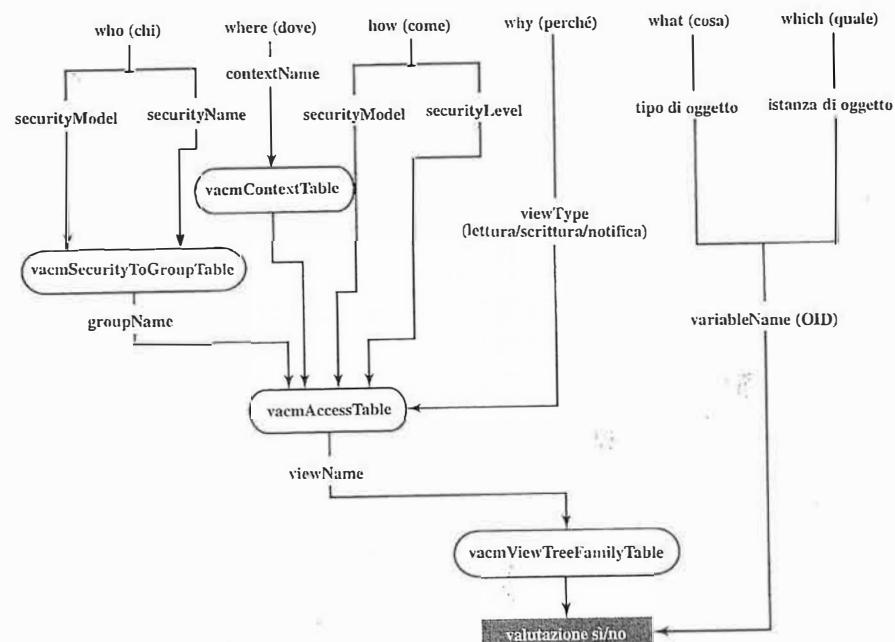


Figura 8.12 Schema logico di VACM.

- **how** (come): la combinazione dei valori di securityModel e securityLevel specifica come la richiesta di accesso o la PDU Inform sono state protette. I valori assunti dalle variabili who, where e how identificano zero o più voci nella tabella vacmAccessTable.
- **why** (perché): il valore di viewType specifica la modalità di accesso richiesta, cioè se la richiesta di accesso è relativa a un'operazione di lettura, scrittura o notifica. La voce selezionata nella tabella vacmAccessTable contiene un valore per viewName per ciascuna delle tre operazioni precedentemente elencate. Il valore di viewType è utilizzato per selezionare uno specifico valore per viewName. Il valore di viewName serve per selezionare la vista MIB appropriata dalla tabella vacmViewTreeFamilyTable.
- **what** (cosa): il valore variableName è un identificatore di oggetto il cui prefisso denota uno specifico tipo di oggetto e il cui suffisso identifica una specifica istanza di oggetto. Il tipo di oggetto identifica la tipologia di informazione di gestione che è stata richiesta.
- **which** (quale): l'istanza di oggetto identifica la specifica informazione richiesta.

Infine, il valore di variableName è confrontato con la vista MIB restituita. L'accesso è consentito se il valore di variableName corrisponde a un elemento della vista.

Motivazioni

I concetti alla base del modello VACM sembrano definire una forma di controllo dell'accesso piuttosto complessa. Le motivazioni di questi concetti sono: da un lato la necessità di chiarire le relazioni coinvolte nell'accedere all'informazione di gestione; dall'altro la volontà di minimizzare le risorse di memorizzazione e di elaborazione presso l'agente. Per meglio comprendere la ragione di queste motivazioni, si osservi come, nel caso di SNMPv1, il concetto di comunità sia utilizzato per rappresentare le seguenti informazioni connesse alla sicurezza:

- l'identità dell'entità richiedente (una stazione di gestione);
- l'identità dell'entità che gestisce la richiesta (un agente che opera in proprio o per conto di un'entità di cui è proxy);
- l'identità dell'entità in cui risiede l'informazione di gestione cui si deve accedere (un agente o un entità gestita tramite proxy);
- informazioni di autenticazione;
- informazioni di controllo dell'accesso (l'autorizzazione per effettuare l'operazione richiesta);
- informazioni sulla vista MIB.

Se tutte queste informazioni fossero fornite mediante un'unica variabile, si perderebbe flessibilità e funzionalità. Di conseguenza, VACM fornisce tutte queste informazioni utilizzando variabili distinte, una per ogni tipologia di informazione. Tale caratteristica rappresenta un sostanziale miglioramento rispetto a SNMPv1, in quanto mantiene separati i vari concetti in modo che a ciascuno possa essere separatamente attribuito un valore.

8.4 Letture consigliate e siti web

[STAL99] fornisce una trattazione completa e dettagliata di SNMP, SNMPv2 e SNMPv3; il libro contiene inoltre una panoramica sulle tecnologie di gestione di rete.

Siti web consigliati

- **Sito web SNMPv3**: gestito dalla Technical University di Braunschweig, fornisce collegamenti ai documenti RFC e ai documenti Internet draft, spiegazioni e modifiche proposte, emesse dal gruppo di lavoro e informazioni sui fornitori di implementazioni SNMPv3.
- **The Simple Web site**: gestito dall'Università di Twente, costituisce una valida sorgente di informazioni su SNMP, inclusi riferimenti a implementazioni public-domain. Il sito fornisce inoltre una lista di libri e articoli sull'argomento.

8.5 Domande di revisione ed esercizi

Domande di revisione

- 8.1 In che senso un'architettura di gestione di rete viene considerata integrata?
- 8.2 Quali sono gli elementi chiave del modello SNMP?
- 8.3 Che cos'è una MIB?
- 8.4 Quali capacità o comandi di base sono forniti in SNMPv1?
- 8.5 Qual è la funzione di un proxy SNMP?
- 8.6 Spiegate brevemente il concetto di comunità di SNMPv1.
- 8.7 Qual è la relazione tra SNMPv1, SNMPv2 e SNMPv3?
- 8.8 Per contrastare quale minaccia è stato progettato USM?
- 8.9 Qual è la differenza tra un motore con autorità e uno senza autorità?
- 8.10 Che cos'è la localizzazione delle chiavi?
- 8.11 Elencare e definire brevemente gli elementi che costituiscono VACM.

Esercizi

- 8.1 In SNMPv1 è definito un tipo di dato chiamato *gauge*, cui viene attribuito il seguente significato.
Questo tipo rappresenta un intero non negativo, che può crescere o decrescere, ma la cui crescita è limitata da un valore massimo. Nello standard viene specificato un valore massimo pari a $2^{32} - 1$ (4294967295).
Sfortunatamente, la definizione di questo valore massimo presenta ambiguità all'interno dello standard. Lo standard SNMPv2 supera tale problema, fornendo la seguente definizione.

Il valore di tipo gauge raggiunge il suo valore massimo quando l'informazione da rappresentare è maggiore o uguale a tale valore massimo; se successivamente l'informazione da rappresentare scende al di sotto del valore massimo, anche il valore di tipo gauge decresce di conseguenza.

- Quale potrebbe essere un'interpretazione alternativa?
- Discutete i pregi e i difetti di entrambe le interpretazioni.

- 8.2** In SNMPv1, a ciascun oggetto in una MIB è associata una categoria di accesso che può assumere uno dei seguenti valori: read-only (sola lettura), read-write (lettura e scrittura), write-only (sola scrittura) e not-accessible (non accessibile). Un'operazione di lettura viene effettuata tramite un'operazione get o trap, mentre un'operazione di scrittura è realizzata mediante un'operazione set. Nel caso di operazioni di sola scrittura, l'oggetto può essere disponibile per operazioni get e trap, in dipendenza dalla specifica implementazione. La categoria di accesso MIB specifica il tipo di accesso più vasto che può essere consentito su un certo oggetto. Tramite la funzionalità della comunità fornita da SNMPv1 è però possibile restringere ulteriormente la modalità di accesso per un certo profilo di comunità. Completate le varie voci della tabella, mostrando gli accessi consentiti.

Categoria di accesso	MIB Modalità di accesso SNMP	
	READ-ONLY	READ-WRITE
read-only		
read-write		
write-only		
not-accessible		

- 8.3** a. Il documento RFC 2574 stabilisce che, nel caso di motori senza autorità, i valori di msgAuthoritativeEngineBoots e msgAuthoritativeEngineTime, contenuti nell'intestazione di un messaggio in uscita, sono inizializzati solo nel caso in cui il messaggio debba essere autenticato dal ricevente con autorità. Spiegate le motivazioni alla base di questa restrizione.

- b. Per contro, nel caso di un messaggio di risposta inviato da un motore con autorità, i valori di msgAuthoritativeEngineBoots e msgAuthoritativeEngineTime vengono sempre inizializzati all'interno dell'intestazione del messaggio in uscita. Motivate tale scelta.

- 8.4** Il documento RFC 2574 specifica che la sincronizzazione dei clock (cioè l'aggiornamento del clock locale sulla base dei valori ricevuti) deve avvenire prima della verifica della finestra temporale (cioè della verifica della tempestività del messaggio). Ciò significa che un motore senza autorità può aggiornare il valore stimato per il clock del motore con autorità se il messaggio ricevuto è autentico, anche se non è contenuto nella finestra temporale. Già dalla pubblicazione del documento RFC 2574, questa possibilità ha suscitato contrasti che hanno trovato spazio sulla mailing list relativa a SNMPv3, ma è chiaro che la dicitura nello standard non cambierà. È utile considerare ciò che questo comporta. Si considerino le seguenti definizioni:

MAEB = msgAuthoritativeEngineBoots

MAET = msgAuthoritativeEngineTime

SEB = stima locale del valore di snmpEngineBoots per il motore remoto con autorità

SET = stima locale del valore di snmpEngineTime per il motore remoto con autorità

LRET = latestReceivedEngineTime

Supponete ora che un motore senza autorità riceva un messaggio tale che:

$$(MAEB = SEB) \text{ AND } [LRET < MAET < (SET - 150)]$$

Le condizioni che consentono di aggiornare il clock sono verificate, e quindi:

$$\text{SET} := \text{MAET}; \quad \text{LRET} := \text{MAET}$$

Se si passa ora alla verifica della finestra temporale si ha:

$$(MAEB = SEB) \text{ AND } (\text{MAET} = \text{SET})$$

e quindi il messaggio può essere considerato esente da ritardi. Si supponga ora che la verifica della finestra temporale sia avvenuta per prima. Indicate se si sarebbe giunti alla stessa conclusione.

- 8.5** Nella versione originaria della specifica USM (documento RFC 2274), è contenuto il seguente commento relativo alle procedure di sincronizzazione dei clock e di verifica della finestra temporale: "Si noti come tale procedura non consenta una sincronizzazione temporale automatica nel caso in cui il motore SNMP senza autorità abbia un'effettiva situazione di desincronizzazione, cioè una situazione in cui il motore SNMP senza autorità ha un ritardo di più di 150 secondi rispetto al motore SNMP con autorità." Tale frase è stata eliminata dalla versione rivista della specifica (documento RFC 2574) dopo che l'autore di questo libro segnalò al gruppo di lavoro che esistono delle situazioni in cui questa frase non è sempre vera. Dimostrate la correttezza di tale affermazione, utilizzando l'esempio fornito nell'Esercizio 8.4.

- 8.6** SNMPv3 si basa sull'ipotesi che esista un qualche mezzo sicuro per trasmettere ai sistemi autenticati (gli agenti) chiavi localized. Le modalità con cui avviene questa trasmissione vanno oltre gli scopi di SNMPv3; la trasmissione delle chiavi può avvenire manualmente oppure tramite un qualche protocollo sicuro. Una volta che una prima chiave (o una coppia, nel caso si voglia assicurare autenticazione e riservatezza) è stata consegnata a un agente, SNMPv3 mette a disposizione un meccanismo per effettuare l'aggiornamento delle chiavi in modo sicuro. È auspicabile che le chiavi vengano di tanto in tanto sostituite, per aumentare la sicurezza. Un utente potrebbe iniziare personalmente il processo di sostituzione delle chiavi, richiedendo e fornendo una nuova password. In alternativa, NMS potrebbe iniziare il processo di sostituzione richiedendo una nuova password. In entrambi i casi, la chiave dell'utente per NMS viene aggiornata. A questo punto NMS può generare una chiave localized per ogni agente in comunicazione. NMS deve quindi comunicare in modalità sicura con ciascun agente per fare in modo che quest'ultimo aggiorni la propria chiave localized. Chiaramente, NMS non può semplicemente inviare la chiave in chiaro attraverso la rete. Per tale operazione esistono due opzioni.

- Cifrare la nuova chiave utilizzando come chiave di cifratura la vecchia chiave.
- Utilizzare un qualche tipo di funzione unidirezionale per generare un valore dalla vecchia chiave. Questo valore è composto mediante l'operazione di OR esclusivo con la nuova chiave. Il risultato è inviato all'agente. L'agente può a questo punto calcolare la nuova chiave effettuando l'operazione di OR esclusivo tra il valore ricevuto e il risultato della stessa funzione unidirezionale applicata alla vecchia chiave.

SNMPv3 utilizza una variante del secondo metodo. Illustrate il vantaggio di questo approccio rispetto al primo.

- 8.7 L'approccio adottato da SNMPv3 implica l'utilizzo di un oggetto di tipo KeyChange nella MIB del sistema contenente la chiave. Un principal remoto o un NMS inizializza tale oggetto, che viene successivamente utilizzato dall'agente in modo automatico per aggiornare la chiave corrispondente. L'algoritmo è suddiviso in due fasi: una di queste viene eseguita presso il motore che invia la richiesta, mentre l'altra è effettuata presso il motore dell'agente remoto. L'intero processo inizia quando qualcuno desidera modificare una chiave esistente, keyOld, facendole assumere un nuovo valore, keyNew. L'entità che inoltra la richiesta effettua i seguenti passi.

1. Generazione di un valore casuale mediante un generatore di numeri pseudo-casuali o mediante un generatore di numeri casuali veri e propri.
2. Calcolo di

$$\text{digest} = \text{Hash}(\text{keyOld} \parallel \text{valore casuale})$$

dove Hash denota MD5 o SHA-1, in base al fatto che si voglia ottenere una chiave di 16 o 20 byte, mentre il simbolo \parallel denota l'operatore di concatenazione.

3. Calcolo di
- $$\text{delta} = \text{digest} \oplus \text{keyNew}$$
- $$\text{protocolKeyChange} = (\text{valore casuale} \parallel \text{delta})$$
- dove il simbolo \oplus denota l'operazione di OR esclusivo.
4. Dopo aver effettuato tali operazioni, il valore di protocolKeyChange viene inviato all'agente mediante un comando Set, al fine di aggiornare un'istanza dell'oggetto KeyChange nella MIB dell'agente.

Illustrate le operazioni che l'agente deve effettuare sul valore ricevuto per aggiornare la chiave.

- 8.8 Un metodo più semplice di quello illustrato nell'esercizio precedente consisterebbe nell'effettuare l'OR esclusivo tra i valori di keyOld e keyNew, e inviare il valore risultante all'agente. L'agente effettuerà quindi l'OR esclusivo tra il valore che ha ricevuto e il valore di keyOld, al fine di generare keyNew. Dal momento che l'avversario non conosce il valore di keyOld, non potrà neanche dedurre il valore di keyNew. Spiegate il vantaggio di utilizzare il numero casuale e la funzione unidirezionale sicura dell'Esercizio 8.7 rispetto all'approccio appena descritto.



Sicurezza di sistema

Questa parte è dedicata ad argomenti di sicurezza a livello sistema, con riferimento alle minacce e alle relative contromisure legate a intrusioni e virus, e all'uso di firewall e sistemi fidati.

Capitolo 9 Intrusioni

Il Capitolo 9 esamina una gamma di possibili accessi alle informazioni e minacce ai servizi rappresentate da hacker o da programmi progettati per sfruttare i punti deboli nei sistemi di elaborazione basati su reti. La trattazione inizia con una discussione delle tipologie di attacchi che possono essere perpetrati da utenti non autorizzati o intrusi, e passa poi ad analizzare i diversi approcci per la prevenzione e la rilevazione; sono poi affrontati temi legati alla gestione delle password.

Capitolo 10 Software dannoso

Il Capitolo 10 esamina le minacce software ai sistemi con un'enfasi particolare su virus e worm. Si inizia con una rassegna dei vari tipi di software dannoso, rivolgendo uno sguardo più dettagliato alla natura di virus e worm. In seguito sono esposte le contromisure ed infine vengono trattati gli attacchi distribuiti di negazione del servizio.

Capitolo 11 Firewall

L'uso dei firewall è un approccio standard alla protezione delle risorse dei computer locali da minacce esterne. In questa parte del libro vengono analizzati i principi generali da seguire nel progetto di un firewall ed esaminate le tecniche specifiche. Inoltre vengono esposti aspetti concernenti i sistemi fidati.

Capitolo 9

Intrusioni

Utenti o programmi software che accedono abusivamente, o quantomeno in maniera indesiderata, ai sistemi connessi in rete rappresentano un'importante problema di sicurezza. L'accesso abusivo da parte di utenti può manifestarsi sotto forma di una richiesta non autorizzata di collegamento (*logon*) a una macchina oppure, in caso di utenti autorizzati, attraverso l'acquisizione di privilegi o l'esecuzione di operazioni al di fuori di quelle autorizzate. L'accesso abusivo da parte di programmi software può manifestarsi come virus, worm o cavalli di Troia.

Tutte queste tipologie di attacco sono connesse alla sicurezza di rete in quanto l'accesso al sistema può essere ottenuto attraverso una rete, anche se non si limitano ad essere perpetrati mediante questa. Un utente con accesso a un terminale locale può tentare di accedere abusivamente al sistema senza sfruttare una rete intermedia. Un virus o un cavallo di Troia possono essere introdotti in un sistema attraverso un dischetto. Solamente il cosiddetto worm rappresenta un fenomeno esclusivamente di rete. Di conseguenza, l'accesso abusivo ai sistemi di elaborazione interessa sia la sicurezza dei sistemi di elaborazione sia la sicurezza di rete.

Dato che questo libro si focalizza sulla sicurezza di rete, non verrà intrapresa un'analisi completa degli attacchi e delle contromisure relativi all'accesso abusivo ai sistemi di elaborazione. Il capitolo inizierà prendendo in esame le intrusioni: in primo luogo verrà considerata la natura degli attacchi e successivamente saranno esaminate le strategie per la prevenzione e – qualora queste fallissero – per la rilevazione. Sarà infine trattata la gestione delle password.

9.1 Intrusioni

Una delle più note minacce alla sicurezza è costituita dagli intrusi (l'altra è rappresentata dai virus), generalmente conosciuti come hacker o cracker (pirati informatici). In uno dei primi studi sulle intrusioni, Anderson [ANDE80], identificò tre categorie di intrusi.

- **Utente masquerade** (*masquerader*): individuo non autorizzato all'uso del sistema che penetra i meccanismi di controllo dell'accesso allo scopo di sfruttare l'account di un utente legittimo.
- **Utente malintenzionato** (*misfeasor*): utente legittimo che accede a dati, programmi o risorse per i quali non possiede autorizzazioni all'accesso oppure possiede autorizzazioni di accesso ma abusa dei propri privilegi.

- **Utente clandestino (*clandestine*)**: individuo che si impadronisce del controllo per la supervisione del sistema e lo utilizza per eludere i meccanismi di controllo dell'accesso e di audit, oppure per nascondere/eliminare l'insieme dei record di audit del sistema.

È plausibile che l'utente masquerade sia un utente esterno sconosciuto al sistema; quello malintenzionato è generalmente un utente interno, noto al sistema, mentre quello clandestino può essere sia un utente esterno sia interno.

Gli attacchi perpetrati dagli intrusi possono essere di tipo benevolo o avere serie conseguenze. Si considerano benevoli gli attacchi derivanti da utenti che sono semplicemente interessati ad esplorare le reti per prendere visione dei contenuti. Sono considerati seri gli attacchi perpetrati al fine di leggere dati privilegiati, effettuare modifiche non autorizzate ai dati o minare il funzionamento del sistema.

La minaccia di un intruso è ben nota grazie al famoso incidente "Wily Hacker" del 1986-87, documentato da Cliff Stoll [STOL88,89]. Nel 1990, negli Stati Uniti, si è verificata una diffusa azione di repressione nei confronti dei pirati informatici con arresti, incriminazioni, molte condanne e la confisca di ingenti quantitativi di dati e dispositivi informatici [STER92]. Furono in molti a credere che il problema fosse sotto controllo.

In realtà non era così. A titolo di esempio, si ricorda che presso i Bell Labs [BELL92, BELL93] si registrarono persistenti e frequenti attacchi sui calcolatori, perpetrati attraverso Internet per un periodo di tempo prolungato e provenienti da svariate sorgenti. All'epoca di questi fatti, i Bell Labs furono oggetto di:

- tentativi di copiare il file delle password con frequenza superiore a una volta ogni due giorni;
- richieste sospette di invocazione di procedura remota (RPC, *remote procedure call*) con frequenza superiore a una volta alla settimana;
- tentativi di connessione a macchine "esca" inesistenti almeno una volta ogni due settimane.

Intrusioni di tipo benevolo possono essere tollerate, anche se consumano risorse di elaborazione e possono rallentare le prestazioni del sistema nei confronti degli utenti legittimi. Non c'è modo tuttavia di conoscere in anticipo se un'intrusione risulterà benevola o maligna. Per tali ragioni, anche per sistemi senza risorse particolarmente sensibili, il problema va tenuto sotto controllo.

Ecco un esempio che illustra una minaccia verificatasi presso la Texas A&M University [SAFF93]. Nell'agosto 1992, il centro di elaborazione ricevette una segnalazione che una delle sue macchine era in procinto di essere usata per effettuare attacchi ad altri calcolatori localizzati altrove, sfruttando Internet. Monitorando le attività, il personale del centro di elaborazione apprese che erano coinvolti parecchi intrusi esterni che eseguivano procedure per la ricostruzione di password (*password-cracking*) su vari calcolatori (il centro era composto da 12.000 macchine interconnesse). Il centro effettuò la disconnessione delle macchine coinvolte, fece fronte alle lacune di sicurezza di cui era a conoscenza e riprese il normale funzionamento. Pochi giorni dopo, uno dei gestori dei sistemi locali scoprì che le intrusioni erano nascoste. Si trovarono file contenenti centinaia di password scoperte, fra cui alcune rela-

tive ai principali server ritenuti sicuri. Inoltre, una macchina locale era stata configurata come BBS per gli hacker e utilizzata da questi per contattarsi reciprocamente e per discutere le tecniche e i progressi raggiunti.

Un'analisi di questo attacco rivelò l'esistenza di due livelli di hacker. Gli hacker di alto livello (le menti) erano utenti molto esperti in possesso di conoscenze approfondite della tecnologia, mentre gli hacker di basso livello (il braccio) svolgevano il ruolo di meri esecutori, addetti al semplice utilizzo dei programmi di cracking, senza comprendere appieno il significato di ciò che stavano eseguendo. Questo lavoro di squadra fu in grado di coniugare le due armi più pericolose a disposizione degli intrusi: da un lato conoscenze sofisticate sulle modalità di intrusione, dall'altro la buona volontà nel dedicare innumerevoli ore ad effettuare tentativi di penetrazione per sondare i punti deboli del sistema.

La crescente consapevolezza del problema delle intrusioni ha portato alla costituzione di gruppi di risposta alle emergenze di sicurezza dei sistemi di elaborazione (CERT, *Computer Emergency Response Team*). I gruppi CERT raccolgono e distribuiscono ai gestori di sistema informazioni relative alle vulnerabilità dei sistemi. Sfortunatamente, anche gli hacker possono avere accesso alle informazioni contenute nei rapporti del CERT. Nel caso dell'incidente presso la Texas A&M, la successiva analisi dei fatti rivelò che gli hacker avevano sviluppato programmi per testare le macchine attaccate rispetto alle diverse vulnerabilità evidenziate dal CERT. Se anche su una sola macchina non erano state risolte le lacune segnalate dal CERT, questa era vittima di attacchi.

Oltre all'esecuzione di programmi di cracking, gli intrusi tentarono di modificare il software preposto all'apertura di sessioni per riuscire a catturare le password degli utenti che si collegavano al sistema. Questo fece in modo che riuscissero ad acquisire un insieme impressionante di password compromesse, che fu collocato nel BBS costituito su una delle macchine dell'organizzazione vittima dell'attacco.

Tecniche di intrusione

L'obiettivo di un intruso è quello di avere accesso a un sistema o di aumentare l'insieme dei privilegi disponibili nel sistema. In generale, ciò richiede che l'intruso entri in possesso di informazioni che avrebbero dovuto restare protette. Nella maggior parte dei casi, si tratta di una password utente. Una volta conosciuta la password utente, l'intruso può aprire una sessione nel sistema ed esercitare tutti i privilegi accordati all'utente legittimo.

Generalmente, un sistema mantiene un file in cui le password sono associate ai rispettivi utenti legittimi. Se il file delle password fosse memorizzato senza alcuna protezione, sarebbe semplice ottenerne l'accesso ed apprendere le password. Il file delle password può essere protetto secondo una delle seguenti modalità.

- **Cifratura unidirezionale (*one-way*)**. Il sistema memorizza solo le password in forma cifrata. Quando un utente fornisce la propria password al sistema, quest'ultimo ne effettua la cifratura e confronta il risultato ottenuto con il valore memorizzato nel file delle password. In pratica, il sistema esegue una trasformazione unidirezionale (ovvero non reversibile) in cui la password è utilizzata per generare la chiave di cifratura e in cui il risultato prodotto in uscita ha una lunghezza fissa.
- **Controllo dell'accesso (*access control*)**. L'accesso al file delle password è limitato a un solo account o un numero molto ristretto di account.

Adottando una o entrambe le misure di protezione, l'acquisizione delle password da parte di un potenziale intruso richiede un certo sforzo. In base a un documento di rassegna della letteratura e a interviste con un certo numero di cracker, [ALVA90] riporta le seguenti tecniche per l'acquisizione di password.

1. Provare password di default utilizzate con account standard fornite con il sistema. Molti amministratori non si preoccupano di cambiare queste password di default.
2. Provare esaustivamente tutte le password di lunghezza limitata (quelle composte da uno a tre caratteri).
3. Provare parole contenute nel dizionario in linea del sistema oppure una lista di password probabili. Esempi di queste ultime sono facilmente disponibili sui BBS degli hacker.
4. Raccogliere informazioni sugli utenti come, ad esempio, i loro nomi, quelli del/della consorte e dei figli, i libri relativi a hobby e i dipinti presenti in ufficio.
5. Provare i numeri di telefono degli utenti, i codici fiscali, i numeri di stanza.
6. Provare tutti i numeri di targa legittimi per lo stato relativo agli utenti in questione.
7. Utilizzare un cavallo di Troia (Paragrafo 9.2) per aggirare i controlli d'accesso.
8. Sfruttare la linea fra un utente remoto e il sistema host.

I primi sei punti mostrano diverse modalità con cui indovinare possibili password valide. Se un intruso dovesse verificare ogni password congetturata mediante una richiesta di connessione (*login*), si genererebbe un attacco noioso e facilmente contrastabile. Ad esempio, il sistema potrebbe semplicemente rifiutare qualunque richiesta di apertura di sessione dopo tre tentativi con password non valide, obbligando l'intruso a riconnettersi all'host per riprovare nuovamente con altre password.

In queste circostanze, diventa impraticabile tentare più di un numero esiguo di password. Tuttavia è improbabile che l'intruso ricorra a questi metodi così grossolani. Ad esempio, se un intruso con limitati privilegi riuscisse a ottenere l'accesso a un file delle password cifrato, la sua strategia consisterebbe nel catturare il file e nell'utilizzare comodamente il meccanismo di cifratura di quel sistema, fino a scoprire una password valida con maggiori privilegi.

Gli attacchi attuati per scoprire le password diventano fattibili e altamente efficaci quando è possibile tentare automaticamente e verificare un numero elevato di password, senza che il processo in corso sia rilevabile. Più avanti saranno ampiamente trattate le modalità con cui contrastare gli attacchi rivolti alla scoperta di password.

Il settimo punto del precedente elenco, relativo al cavallo di Troia, descrive una modalità particolarmente difficile da contrastare. Un esempio di programma che è riuscito ad aggirare i controlli dell'accesso è descritto in [ALVA90]. Un utente con limitati privilegi dopo aver creato un programma gioco, ha invitato l'operatore di sistema a utilizzarlo nel suo tempo libero. In effetti, il programma eseguiva un gioco ma conteneva anche codice nascosto per copiare, in un file dell'utente in questione, il file delle password (che era memorizzato in chiaro ma protetto mediante un meccanismo di controllo degli accessi). Il programma gioco era in grado di accedere al file delle password, in quanto era eseguito dall'operatore e quindi operava in modalità privilegiata.

L'ottavo punto dell'elenco, relativo all'attacco di sfruttamento della linea, riguarda la sicurezza fisica. Può essere contrastato mediante tecniche di cifratura a livello linea, trattate nel Paragrafo 7.1.

Altre tecniche di intrusione non richiedono la conoscenza di una password: gli intrusi possono entrare in un sistema sfruttando attacchi del tipo dell'overflow del buffer su un programma in esecuzione con certi privilegi. In questo modo chi vuole effettuare un'intrusione può accedere al sistema con privilegi sempre più elevati.

Si riprende ora la discussione delle due principali contromisure nei confronti delle intrusioni: rilevazione e prevenzione. La rilevazione riguarda la scoperta di un attacco, prima o dopo che questa abbia successo. La prevenzione rappresenta invece una sfida, essendo un obiettivo difficile da raggiungere. La difficoltà deriva dal fatto che per difendersi occorre contrastare tutti i possibili attacchi, mentre l'avversario è libero di identificare l'anello più debole nella catena delle difese e sfruttarlo per attaccare proprio in quel punto.

9.2 Rilevazione delle intrusioni

Ovviamente, anche il miglior sistema di prevenzione delle intrusioni può fallire. La seconda linea di difesa consiste nella rilevazione delle intrusioni, su cui si è concentrata molta attività di ricerca negli ultimi anni. Le motivazioni per tale interesse possono essere riassunte come segue.

1. Se un'intrusione viene rilevata con sufficiente rapidità, l'intruso può essere identificato ed espulso dal sistema prima che possa procurare danni o compromettere i dati. Anche se la rilevazione non avviene in tempo utile per anticipare l'intruso, quanto più velocemente si riesce a rilevare l'intrusione tanto minore sarà l'entità dei danni arrecati e tanto più velocemente si attuerà il ripristino.
2. Un efficace sistema di rilevazione delle intrusioni può servire da deterrente, agendo quindi come prevenzione delle intrusioni.
3. La rilevazione delle intrusioni permette di raccogliere informazioni relative alle tecniche di intrusione, informazioni utilizzabili per rafforzare e migliorare le funzionalità di prevenzione.

La rilevazione delle intrusioni si basa sull'ipotesi che il comportamento degli intrusi differisce da quello degli utenti legittimi in maniera quantificabile. Naturalmente, non ci si può aspettare una differenza marcata ed esatta fra un attacco perpetrato da un avversario e il normale utilizzo delle risorse di sistema da parte di un utente autorizzato. Piuttosto, ci si deve aspettare qualche sovrapposizione.

La Figura 9.1 suggerisce, in termini molto astratti, la natura del compito affrontato dal progettista di un sistema di rilevazione delle intrusioni. Anche se il comportamento tipico di un intruso differisce dal comportamento tipico di un utente autorizzato, fra i due comportamenti c'è una certa sovrapposizione. Pertanto, un'interpretazione meno restrittiva del comportamento di un intruso finalizzata a catturare molte intrusioni tenderà anche a creare un certo numero di falsi positivi, o utenti autorizzati riconosciuti come intrusi. D'altro canto, il tentativo di limitare i falsi positivi mediante un'interpretazione restrittiva del com-

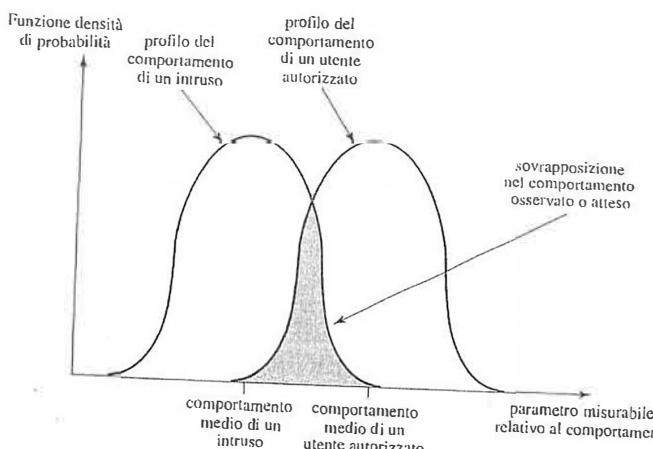


Figura 9.1 Profili di comportamento per intrusi e utenti autorizzati.

portamento di un intruso porterà ad avere un aumento dei falsi negativi, ovvero di intrusi non identificati come tali. Pertanto, la pratica dei sistemi di rilevazione richiede una certa dose di compromessi e abilità.

In uno studio di Anderson [ANDE80] si postulava che è possibile distinguere, con un ragionevole livello di confidenza, un utente masquerade da uno legittimo. Si possono stabilire le tipologie di comportamento degli utenti legittimi osservando la storia e si possono rilevare deviazioni significative da tali tipologie. Anderson suggeriva che l'identificazione di un utente malintenzionato (ovvero un utente legittimo che si comporta in maniera non autorizzata) è più complessa, in quanto la distinzione fra comportamento normale e anomalo potrebbe essere minima. Anderson concludeva che tali violazioni non sarebbero rilevabili unicamente mediante la ricerca di comportamenti anomali. Ciò nonostante, il comportamento di utenti malintenzionati è rilevabile definendo in maniera intelligente le classi di condizioni che denotano un uso non autorizzato delle risorse del sistema. Infine, la rilevazione di utenti clandestini veniva riconosciuta come non inherente all'ambito delle tecniche di tipo totalmente automatico. Queste considerazioni, che risalgono al 1980, sono tutt'oggi valide. [PORR92] identifica i seguenti approcci per la rilevazione di intrusioni.

1. **Rilevazione statistica delle anomalie:** comporta la raccolta di dati relativi al comportamento di utenti legittimi in un determinato periodo di tempo. Sul comportamento osservato si eseguono test statistici per determinare con livello di confidenza elevato se quel comportamento è da considerarsi illegittimo.
 - a. Individuazione delle soglie: questo approccio comporta la definizione di soglie, indipendentemente dagli utenti, per la frequenza con cui si verificano i vari eventi.
 - b. Sistemi basati su profili: si sviluppa un profilo dell'attività di ciascun utente, che viene utilizzato per individuare cambiamenti nel comportamento relativo a ciascun account.

2. **Rilevazione basata su regole:** comporta la definizione di un insieme di regole che possono essere utilizzate per decidere che un dato comportamento è quello di un intruso.
 - a. Rilevazione di anomalie: si sviluppano regole per individuare deviazioni rispetto a tipologie di utilizzo precedentemente definite.
 - b. Identificazione di penetrazioni: approccio basato su sistemi esperti che ricercano comportamenti sospetti.

In sintesi, gli approcci statistici tentano di definire il comportamento considerato normale o atteso, mentre gli approcci basati su regole tentano di definire il comportamento corretto.

Rispetto alle tipologie di attacchi precedentemente elencate, la rilevazione statistica di anomalie è efficace contro attacchi di utenti masquerade, che probabilmente non imiteranno le tipologie di comportamento degli account di cui si sono appropriati. D'altro canto, tali tecniche possono risultare inefficaci nel caso di utenti malintenzionati. Per tali attacchi, gli approcci basati su regole possono essere in grado di riconoscere eventi e sequenze che in tale contesto sono sintomatici di una penetrazione. In pratica, un sistema può combinare entrambi gli approcci al fine di risultare efficace rispetto a una vasta gamma di attacchi.

Record di audit

I record di audit costituiscono uno strumento fondamentale per la rilevazione di intrusioni. Come ingresso al sistema di rilevazione delle intrusioni deve essere mantenuta una registrazione delle attività in corso. Fondamentalmente, si utilizzano due tipologie di record di audit.

- **Record di audit nativi:** praticamente tutti i sistemi operativi multiutente contengono software di accounting che raccoglie informazioni sulle attività degli utenti. Utilizzando queste informazioni, si evita l'impiego di ulteriore software di raccolta. Lo svantaggio è che i record nativi possono non contenere le informazioni necessarie oppure possono contenere ma non in formato appropriato.
- **Record di audit specifici per la rilevazione:** si può realizzare una funzionalità di raccolta per la generazione di record di audit contenenti solo le informazioni necessarie al sistema di rilevazione delle intrusioni. Un vantaggio di questo approccio è che può essere reso indipendente dagli specifici ambienti e portatile su una vasta gamma di sistemi. Lo svantaggio è relativo al carico di elaborazione aggiuntivo derivante dall'avere due software di accounting in esecuzione sulla stessa macchina.

Un buon esempio di record di audit specifico per la rilevazione è stato sviluppato da Dorothy Denning [DENN87]. Ciascun record di audit contiene i seguenti campi.

- **Soggetto:** colui che ha iniziato l'azione. Un soggetto tipicamente è un utente connesso mediante terminale ma potrebbe anche essere un processo che agisce per conto di un utente o di un gruppo di utenti. Tutta l'attività è generata dai comandi inviati dai soggetti; questi possono essere raggruppati in differenti classi di accesso che possono anche sovrapporsi.

- Azione:** operazione eseguita dal soggetto su o mediante un oggetto; ad esempio: connessione, lettura, operazione di I/O, esecuzione.
- Oggetto:** ricettore di azioni. Esempi di oggetti comprendono file, programmi, messaggi, record, terminali, stampanti, strutture dati create da utenti e/o programmi. Quando un soggetto agisce da ricettore di un'azione come, ad esempio, la posta elettronica, viene considerato un oggetto. Gli oggetti possono essere raggruppati per tipo. La granularità degli oggetti può variare per tipo e ambiente. Ad esempio, le azioni su basi di dati possono essere registrate a livello di intera base di dati oppure a livello di singolo record.
- Condizione di eccezione:** indica l'eventuale condizione di eccezione restituita al soggetto in caso di azione parzialmente o totalmente rifiutata dal sistema.
- Utilizzo risorse:** una lista di elementi quantitativi in cui ogni elemento fornisce l'ammontare di utilizzo di una certa risorsa (per esempio, numero di linee stampate o visualizzate, numero di record letti o scritti, tempo di CPU, unità di I/O utilizzate, tempo di sessione trascorso).
- Timestamp:** valore univoco di tempo e data, che identifica l'istante in cui si è verificata l'azione.

La maggior parte delle operazioni degli utenti è composta da un certo numero di azioni elementari. Ad esempio, un'operazione di copia di file comporta l'esecuzione del comando di lettura da un file e l'operazione di scrittura sull'altro file. Si consideri il comando

COPY GAME.EXE TO <library>GAME.EXE

eseguito dall'utente Smith per copiare il file eseguibile GAME dalla directory corrente alla directory <Library>. Si possono generare i seguenti record di audit:

Smith	esecuzione	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	lettura	<Smith>GAME.EXE	0	RECORD = 0	11058721679
Smith	esecuzione	<Library>COPY.EXE	violaz. scrittura	RECORD = 0	11058721680

In questo caso, il comando di copia non va a buon fine (è abortito) perché l'utente Smith non ha il permesso di scrittura sulla directory <Library>.

La scomposizione di un'operazione utente in azioni elementari presenta tre vantaggi.

- Dato che gli oggetti rappresentano le entità da proteggere nel sistema, l'uso di azioni elementari rende possibile la registrazione di tutti i comportamenti che interessano un oggetto. Pertanto, il sistema può rilevare tentativi di sabotaggio dei meccanismi di controllo dell'accesso (notando un'anomalia nel numero di condizioni di eccezione restituite) e può rilevare sabotaggi andati a buon fine notando un'anomalia nell'insieme degli oggetti accessibili al soggetto.

- Record di audit su singoli oggetti e singole azioni semplificano il modello e la sua realizzazione.
- Data la struttura semplice e uniforme dei record di audit specifici per la rilevazione, può risultare relativamente semplice ottenere questa informazione o una sua parte, stabilendo una semplice corrispondenza fra i record nativi esistenti e i record specifici per la rilevazione.

Rilevazione statistica delle anomalie

Come accennato in precedenza, le tecniche di rilevazione statistica delle anomalie ricadono in due categorie: identificazione di soglie e sistemi basati su profili. L'identificazione di soglie comporta il conteggio del numero di occorrenze di uno specifico tipo di evento in un dato intervallo di tempo. Se il conteggio supera quello che è considerato un numero ragionevole atteso, allora si ipotizza l'intrusione.

L'analisi delle soglie risulta di per sé uno strumento piuttosto grossolano e inefficace per la rilevazione di attacchi anche moderatamente sofisticati. Devono essere determinati sia i valori di soglia sia l'intervallo di tempo. Data la variabilità sugli utenti, tali valori di soglia probabilmente genereranno un numero elevato o di falsi positivi o di falsi negativi. Semplici rilevatori basati su soglie possono tuttavia risultare utili se combinati con tecniche più sofisticate.

La rilevazione delle anomalie basata su profili si concentra sulla caratterizzazione del comportamento osservato di singoli utenti o di gruppi e sulla rilevazione di deviazioni significative. Un profilo può essere costituito da un insieme di parametri, in modo che la deviazione su un singolo parametro può non essere sufficiente di per sé a segnalare un allarme.

Tale approccio si fonda sull'analisi dei record di audit. Questi forniscono informazioni in ingresso alla funzione di rilevazione in due modi. In primo luogo, il progettista deve decidere un certo numero di metriche quantitative che si possano usare per effettuare la misurazione del comportamento dell'utente. L'analisi dei record di audit per un periodo di tempo può essere usata per determinare il profilo delle attività dell'utente medio. I record di audit servono quindi per la definizione del comportamento tipico. In secondo luogo, i record di audit correnti costituiscono l'ingresso per la rilevazione delle intrusioni. Ovvvero, il modello di rilevazione analizza i record in ingresso per determinare deviazioni dal comportamento medio. Ecco alcuni esempi di metriche utili per la rilevazione basata su profili.

- Contatore:** intero non negativo che può essere incrementato ma non decrementato fino a quando non viene azzerato. Generalmente, si mantiene un contatore di determinati tipi di eventi per un dato periodo di tempo. Esempi di contatori comprendono il numero di operazioni di connessione per singolo utente all'ora, il numero di volte che un certo comando viene eseguito durante una sessione e il numero di tentativi di immissione di password falliti in un minuto.
- Gauge:** intero non negativo che può essere incrementato o decrementato. Generalmente, si usa un gauge per misurare il valore corrente di qualche entità. Esempi di gauge comprendono il numero di connessioni logiche assegnate a un'applicazione utente e il numero di messaggi uscenti in coda per un processo utente.

- Intervallo temporale:** lunghezza dell'intervallo di tempo fra due eventi correlati. Un esempio è la lunghezza dell'intervallo temporale fra due operazioni di connessione successive relative a un certo account.

- Utilizzazione di risorse:** quantità di risorse consumate durante un periodo specificato. Esempi di utilizzazione di risorse sono il numero di pagine stampate durante una sessione utente e il tempo totale consumato dall'esecuzione di un programma.

Date queste metriche generali, si possono eseguire diversi test per determinare se l'attività in corso rientra entro certi limiti considerati accettabili. [DENN87] elenca i seguenti approcci adottabili:

- media e deviazione standard
- approccio multivariato
- processo markoviano
- serie temporali
- modello operazionale.

Il test statistico più semplice consiste nell'effettuare la misurazione della **media** e della **deviazione standard** di un parametro per un dato periodo temporale. Questi valori riflettono il comportamento medio e la sua variabilità. La media e la deviazione standard sono applicabili a una vasta gamma di contatori, misurazioni temporali e misurazioni di risorse. Ma queste misurazioni di per sé sono troppo grossolane per scopi di rilevazione di intrusioni.

Il modello **multivariato** si basa sulle correlazioni fra due o più variabili. Il comportamento degli intrusi può essere caratterizzato con un livello di confidenza più elevato considerando tali correlazioni (ad esempio: il tempo di elaborazione e l'utilizzazione delle risorse, la frequenza delle connessioni e il tempo di sessione trascorso).

Si utilizza un modello basato sul **processo markoviano** per stabilire la probabilità di transizione fra vari stati. Ad esempio, questo modello potrebbe essere utilizzato per analizzare transizioni fra alcuni comandi.

Un modello basato sulle **serie temporali** si focalizza su intervalli di tempo, cercando sequenze di eventi che si verificano troppo rapidamente o troppo lentamente. Si possono applicare una vasta gamma di test statistici per caratterizzare anomalie temporali.

Infine, un **modello operazionale** si basa sulla valutazione di ciò che è ritenuto anomalo, piuttosto che sull'analisi automatizzata dei record di audit precedenti. Tipicamente, si stabiliscono dei limiti fissati e, in corrispondenza di un'osservazione al di fuori di questi, si sospetta un'intrusione. Questo approccio funziona meglio in contesti in cui il comportamento degli intrusi può essere dedotto da certe tipologie di attività. Ad esempio, un elevato numero di tentativi di connessione in un periodo di tempo limitato suggerisce un tentativo di intrusione.

A titolo di esempio sull'uso delle diverse metriche e dei diversi modelli, la Tabella 9.1 mostra le misure considerate o testate per il sistema di rilevazione delle intrusioni (denominato IDES, *intrusion detection system*) sviluppato presso lo Stanford Research Institute (SRI) [DENN87, JAVI91, LUNT88].

Il vantaggio principale derivante dall'uso dei profili statistici è che non è richiesta conoscenza a priori dei punti deboli per la sicurezza. Il programma di rilevazione apprende in

Misura	Modello	Tipologia di intrusione rilevata
Attività di login e di sessione		
Frequenza di login per giorno e tempo	Media e deviazione standard	Gli intrusi molto probabilmente si collegano al di fuori del normale orario di lavoro.
Frequenza di login da diverse postazioni	Media e deviazione standard	Gli intrusi si possono collegare da una postazione utilizzata raramente o mai da un certo utente.
Tempo trascorso dall'ultima connessione	Operazionale	Attività che sfrutta un account "morto".
Tempo trascorso per sessione	Media e deviazione standard	Deviazioni significative possono essere sintomatiche di utenti masquerade.
Quantità di dati in output a una certa postazione	Media e deviazione standard	Quantità eccessive di dati trasmessi a postazioni remote possono indicare che è in atto un rilascio di dati sensibili.
Utilizzo delle risorse di sessione	Media e deviazione standard	Elaborazioni o livelli di I/O inusuali possono essere sintomatici di una intrusione.
Numero di password errate in base di login	Operazionale	Tentativo di intrusione mediante scoperta di password.
Insuccessi di login da specifici terminali	Operazionale	Tentativo di intrusione.
Attività di esecuzione di programmi o comandi		
Frequenza di esecuzione	Media e deviazione standard	Può scoprire intrusi che probabilmente utilizzano comandi diversi, o una penetrazione nel sistema riuscita da parte di un utente legittimo che ha avuto accesso a comandi privilegiati.
Utilizzo delle risorse da parte dei programmi	Media e deviazione standard	Un valore anomalo può suggerire l'inserimento di un virus o di un cavallo di Troia generando effetti collaterali che aumentano il tasso di utilizzo del processore o dell'I/O.
Negazione delle richieste di esecuzione	Operazionale	Può rilevare tentativi di penetrazione nel sistema da parte di singoli individui che cercano di acquisire maggiori privilegi.
Attività di accesso a file		
Frequenza di lettura, scrittura, cancellazione	Media e deviazione standard	Anomalie di accessi in lettura e scrittura per singoli individui possono indicare attacchi masquerade o di browsing.
Record letti, scritti	Media e deviazione standard	L'anomalia può essere sintomatica di un tentativo di ottenere dati sensibili mediante inferenza e aggregazione.
Conteggio di operazioni di lettura, scrittura, cancellazioni fallite	Operazionale	Possono rilevare utenti che effettuano tentativi di accesso a file creazione e non autorizzati in maniera persistente.

Tabella 9.1 Misure utilizzabili per la rilevazione delle intrusioni.

cosa consiste un comportamento “normale” e quindi ricerca deviazioni. L'approccio non si basa su caratteristiche e vulnerabilità dipendenti da uno specifico sistema. Pertanto, il programma dovrebbe risultare portabile su una vasta gamma di sistemi.

Rilevazione delle intrusioni basata su regole

Le tecniche basate su regole rilevano le intrusioni osservando gli eventi nel sistema e applicando un insieme di regole che portano a decidere se una certa tipologia di attività è da considerarsi sospetta o meno. In termini molto generali, tali tecniche possono essere classificate o come tecniche orientate alla rilevazione di anomalie o come tecniche orientate all'identificazione di penetrazioni, anche se le due categorie possono sovrapporsi.

La **rilevazione di anomalie basata su regole** risulta simile alla rilevazione statistica di anomalie rispetto all'approccio di base e alle sue capacità di rilevazione. Secondo questo approccio, si analizzano record di audit storici al fine di identificare tipologie di utilizzo e generare automaticamente regole per la loro descrizione. Le regole possono rappresentare tipologie di comportamento passate relative a utenti, programmi, privilegi, intervalli temporali, terminali e così via. Si osserva quindi il comportamento corrente e ogni transazione viene confrontata con l'insieme di regole, per valutare se è conforme a qualche tipologia di comportamento di tipo storico.

Come nel caso della rilevazione statistica di anomalie, la rilevazione basata su regole non richiede conoscenza delle vulnerabilità di sicurezza presenti nel sistema. Piuttosto, lo schema si basa sull'osservazione del comportamento passato e sull'ipotesi che il futuro sarà simile al passato. Affinché tale approccio sia efficace, è necessario disporre di una base di dati piuttosto ampia di regole. Ad esempio, nello schema descritto in [VACC89], la base di dati contiene da 10^4 a 10^6 regole.

L'**identificazione delle penetrazioni basate su regole** segue un approccio completamente diverso per la rilevazione di intrusioni, basato sulla tecnologia dei sistemi esperti. La caratteristica peculiare di tali sistemi è l'uso di regole per identificare tipologie di penetrazioni conosciute oppure penetrazioni che potrebbero sfruttare noti punti deboli del sistema. Si possono anche definire regole per l'identificazione di comportamenti sospetti, anche quando il comportamento rientra nei limiti delle tipologie di utilizzo stabilite. Generalmente, le regole utilizzate in questi sistemi sono specifiche per la macchina e il sistema operativo in questione. Inoltre, tali regole sono generate da “esperti” piuttosto che derivate dall'analisi automatizzata dei record di audit. La normale procedura seguita a tale scopo consiste nell'intervistare gli amministratori di sistema e gli analisti di sicurezza al fine di raccogliere un insieme di scenari di penetrazione e relativi eventi chiave che rappresentano una minaccia per il sistema da proteggere¹. La validità dell'approccio dipende quindi dall'esperienza e dalle conoscenze delle persone coinvolte nella definizione delle regole.

Un semplice esempio del tipo di regole che si possono utilizzare si trova nel sistema NIDX, un sistema di vecchia generazione che utilizzava regole euristiche per attribuire un livello di potenziale sospetto alle attività [BAUE88]. Ecco alcuni esempi di euristiche impiegate.

¹ Tali interviste possono coinvolgere anche cracker pentiti o non pentiti, retribuiti per mettere a disposizione le loro competenze [FREE93].

1. Gli utenti non dovrebbero leggere file collocati nelle directory personali di altri utenti.
2. Gli utenti non devono scrivere su file di altri utenti.
3. Gli utenti che si ricongliono al sistema dopo un certo numero di ore, in molti casi accedono agli stessi file che avevano usato in precedenza.
4. Di solito, gli utenti non aprono direttamente i dispositivi disco, ma sfruttano a tale scopo funzionalità di più alto livello del sistema operativo.
5. Gli utenti non dovrebbero essere collegati più di una volta allo stesso sistema.
6. Gli utenti non fanno copie dei programmi di sistema.

Lo schema di identificazione delle penetrazioni utilizzato nel sistema IDES è rappresentativo della strategia seguita. I record di audit sono esaminati al momento della generazione e vengono confrontati con la base delle regole. Se qualche regola è verificata, si incrementa il livello di sospetto (*suspicion rating*) dell'utente. Se il tracciato di audit verifica un certo numero di regole, il livello di sospetto sorpasserà il valore di soglia, generando la segnalazione dell'anomalia.

Azione USTAT	Tipo di evento SunOS
Read (Lettura)	open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt
Write (Scrittura)	truncate, truncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wct
Create (Creazione)	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwlc, open_wc, open_wtc, mknod
Delete (Cancellazione)	rmdir, unlink
Execute (Esecuzione)	exec, execve
Exit (Uscita)	exit
Modify_Owner (Modifica_Proprietario)	chown, fchown
Modify_Perms (Modifica_Permessi)	chmod, fchmod
Rename (Ridenominazione)	rename
Hardlink (Collegamento)	link

Tabella 9.2 Azioni del modello USTAT e corrispondenti tipi di eventi SunOS.

L'approccio seguito in IDES si fonda sull'analisi dei record di audit. Un punto debole di tale approccio è la mancanza di flessibilità. Dato un certo scenario di penetrazione, si possono produrre diverse sequenze alternative di record di audit, ciascuna delle quali varia lievemente rispetto alle altre, magari in maniera impercettibile. Tenere in considerazione tutte queste variazioni attraverso regole esplicite potrebbe risultare complesso. Un altro metodo consiste nel definire un modello di alto livello, indipendente dagli specifici record di audit. Un esempio di questo approccio è il modello a transizioni di stato conosciuto come USTAT [ILGU93]. USTAT considera azioni generali piuttosto che non le specifiche azioni registrate con il meccanismo di audit di UNIX. USTAT è implementato su un sistema SunOS che fornisce record di audit relativamente a 239 eventi. Di questi, solo 28 sono utilizzati da un pre-processore che li trasforma in 10 azioni di tipo generale (Tabella 9.2, a pagina precedente). Utilizzando solo queste azioni e i parametri invocati da ciascuna azione, si costruisce un diagramma di transizione di stato caratterizzante un'attività sospetta. Dato che un certo numero di eventi distinti registrabili vengono messi in corrispondenza con un insieme più ridotto di azioni, il processo di creazione delle regole si semplifica. Inoltre, il diagramma di transizione di stato può essere facilmente modificato per riflettere nuovi comportamenti di intrusione.

La Base-rate Fallacy

Per essere utilizzabile in pratica, un sistema di rilevazione dell'intrusione dovrebbe individuare una sostanziosa percentuale di intrusioni, mentre mantiene il tasso di falsi allarmi a un livello accettabile. Se solo una modesta percentuale delle effettive intrusioni viene individuata, il sistema fornisce un falso senso di sicurezza. D'altra parte se il sistema segnala frequentemente allarmi anche quando non ci sono intrusioni (un falso allarme), allora o i gestori del sistema cominciano a ignorare gli allarmi o si perderà la maggior parte del tempo nell'analisi dei falsi allarmi.

Sfortunatamente, a causa della natura delle probabilità coinvolte, è molto difficile soddisfare lo standard di un elevato tasso di rilevazioni e di un basso tasso di falsi allarmi. In generale, se il numero delle effettive intrusioni è basso, confrontato con il numerosi usi legittimi del sistema, allora il tasso di falsi allarmi sarà alto, a meno che il test sia estremamente discriminante. Uno studio sui sistemi di rilevazione delle intrusioni esistenti, riportato in [AXEL00] ha indicato che i sistemi attuali non hanno superato il problema del base-rate fallacy. Si veda l'Appendice 9A per informazioni concise sulla matematica legata a questo problema.

Rilevazione distribuita delle intrusioni

Fino a poco tempo fa, la ricerca sui sistemi di rilevazione delle intrusioni si era concentrata su singoli sistemi stand-alone. Tuttavia, un'organizzazione ha necessità di proteggere insiemi di host distribuiti, connessi su una LAN o su reti interconnesse fra loro. Anche se è possibile organizzare una protezione utilizzando sistemi di rilevazione delle intrusioni di tipo stand-alone su ciascun host, si può ottenere una protezione più efficace facendo cooperare e coordinando i sistemi di rilevazione delle intrusioni attraverso la rete.

Porras evidenzia le principali problematiche connesse con la progettazione di sistemi di rilevazione distribuita delle intrusioni [PORR92].

- Un sistema per la rilevazione distribuita delle intrusioni può avere necessità di considerare diversi formati di record di audit. In un ambiente eterogeneo, sistemi diversi utilizzano differenti sistemi nativi di audit e, se si fa uso di un sistema di rilevazione delle intrusioni, possono utilizzare formati diversi per i record di audit connessi con la sicurezza.
- Uno o più nodi della rete hanno il ruolo di punti di raccolta e analisi dei dati provenienti dai sistemi in rete. Quindi, esiste la necessità di trasmettere sulla rete dati di audit grezzi o dati di sintesi. Pertanto, occorre assicurare l'integrità e la riservatezza di tali dati. L'integrità è necessaria per impedire a un intruso di mascherare le proprie attività modificando i dati di audit trasmessi. La riservatezza è necessaria perché i dati di audit trasmessi potrebbero essere di grande valore.
- Si può scegliere un'architettura centralizzata o distribuita. Con un'architettura centralizzata, esiste un unico punto centrale per la raccolta e l'analisi dei dati di audit. Ciò facilita il compito di correlare i report inviati ma genera un collo di bottiglia potenziale e un solo punto di guasto. Con un'architettura distribuita, i centri di analisi sono più di uno e devono coordinare le proprie attività e scambiarsi informazioni.

Il sistema sviluppato presso la University of California at Davis rappresenta un buon esempio di sistema per la rilevazione distribuita delle intrusioni [HEBE92, SNAP91]. La Figura 9.2 mostra l'architettura complessiva del sistema che presenta tre componenti principali.

- **Modulo agente dell'host:** modulo di raccolta di dati di audit che opera come processo background su un sistema oggetto di controllo. Il suo scopo è la raccolta di dati su eventi connessi con la sicurezza che accadono sull'host e la trasmissione di tali dati al gestore centrale.

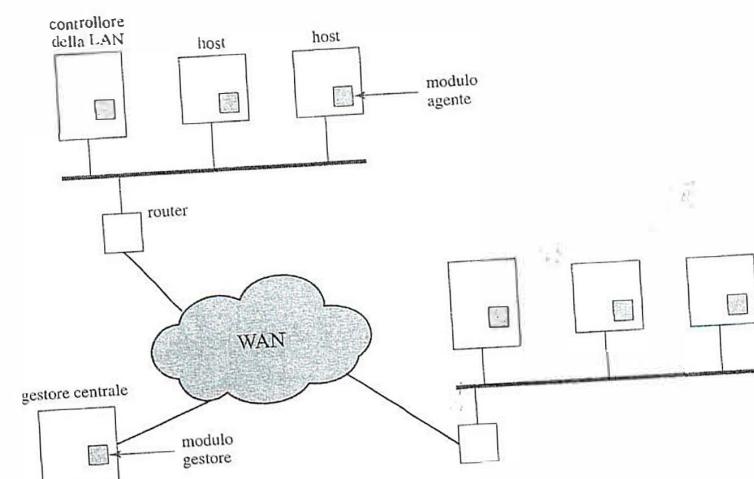


Figura 9.2 Architettura per la rilevazione distribuita delle intrusioni.

- Modulo agente per il controllo della LAN:** opera in maniera analoga a quella del modulo agente dell'host ad eccezione del fatto che analizza il traffico LAN e riporta i risultati al gestore centrale.
- Modulo gestore centrale:** riceve i report dai due agenti precedenti, elaborando e correlando i report ricevuti per la rilevazione delle intrusioni.

Lo schema è progettato per essere indipendente da qualunque sistema operativo o realizzazione del sistema di audit. La Figura 9.3 [SNAP91] mostra l'approccio generale adottato. L'agente cattura ogni record di audit prodotto dal sistema nativo di audit. Viene applicato un filtro che trattiene solo i record di interesse per la sicurezza. Tali record sono riformattati secondo un formato standard, conosciuto come HAR (*host audit record*). Successivamente, un modulo logico basato su template analizza questi record per rilevare attività sospette. A livello più basso, l'agente ricerca eventi significativi di interesse indipendentemente da qualunque altro evento passato. Esempi di tali eventi comprendono accessi falliti a file, accessi a file di sistema e modifiche dei permessi di controllo dell'accesso a tipologie conosciute di attacchi (*signature*). Infine, l'agente ricerca comportamenti anomali di singoli utenti sulla base dei profili storici degli utenti stessi come, ad esempio, il numero di programmi eseguiti, il numero di file cui ha avuto accesso e così via.

Quando si rilevano attività sospette, viene inviato un allarme al gestore centrale. Quest'ultimo contiene un sistema esperto che può eseguire inferenze sui dati ricevuti. Il ge-

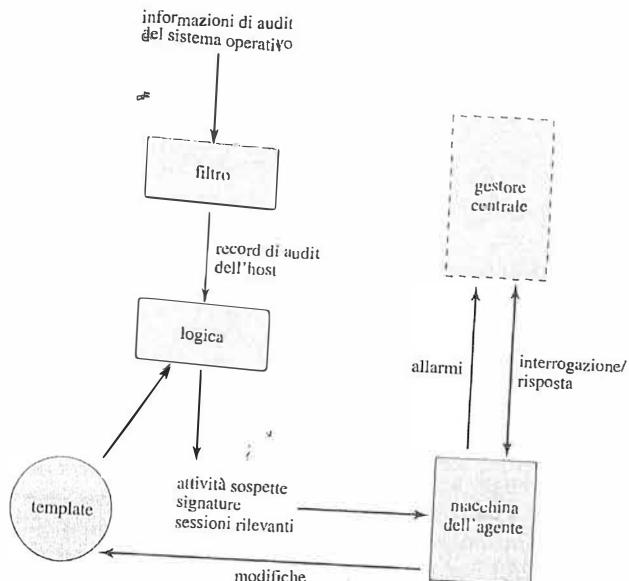


Figura 9.3 Architettura dell'agente.

store può anche richiedere ai singoli sistemi copie dei loro HAR per effettuarne la correlazione con quelli provenienti da altri agenti.

Anche l'agente di controllo della LAN fornisce informazioni al gestore centrale. Tale agente effettua registrazioni di audit relativamente a connessioni tra due host, ai servizi utilizzati e al volume del traffico. Ricerca eventi significativi come improvvisi cambiamenti nel carico della rete, l'utilizzo di servizi relativi alla sicurezza e attività di rete come, ad esempio, comandi di connessione remota (*rlogin*).

L'architettura mostrata nelle Figure 9.2 e 9.3 è abbastanza generale e flessibile. Rappresenta uno schema di base per un approccio indipendente dalla macchina che può espandersi da sistemi di rilevazione delle intrusioni di tipo stand-alone a un sistema in grado di correlare le attività di un certo numero di siti e reti, al fine di identificare attività sospette che in caso contrario resterebbero sconosciute.

Honeypot

Un'innovazione relativamente recente nella tecnologia di rilevazione delle intrusioni è honeypot (letteralmente: vasetto di miele). Gli honeypot sono sistemi che fanno da esca e che sono progettati per allettare un potenziale attaccante ad abbandonare i sistemi critici. Gli honeypot sono progettati per:

- distogliere un attaccante dall'accesso ai sistemi critici;
- raccogliere informazioni sull'attività dell'attaccante;
- incoraggiare l'attaccante a rimanere nel sistema abbastanza a lungo affinché gli amministratori possano reagire.

Questi sistemi sono riempiti con informazioni falsificate, progettate per apparire preziose, ma alle quali un utente legittimo del sistema non avrebbe accesso. Quindi, qualsiasi accesso all'honeypot è sospetto. Questo sistema è dotato di sensibili dispositivi di controllo e di registratori degli eventi che individuano questi accessi e raccolgono informazioni sull'attività dell'attaccante. Dal momento che si fa sembrare che ogni attacco all'honeypot abbia avuto successo, gli amministratori hanno il tempo di mobilitarsi, registrare e seguire le tracce dell'attaccante senza mai esporre i sistemi produttivi.

Gli sforzi iniziali si sono concentrati su un singolo computer honeypot con un indirizzo IP progettato in modo tale da attirare gli hacker. Ricerche più recenti si sono focalizzate sulla costruzione di un'intera rete honeypot che emula un'azienda, possibilmente con dati e traffico effettivi o simulati. Una volta che gli hacker sono all'interno della rete, gli amministratori possono osservarne il comportamento in dettaglio e pensare delle difese.

Formato per lo scambio delle informazioni di rilevazione delle intrusioni

Per facilitare lo sviluppo di sistemi distribuiti per la rilevazione delle intrusioni, che possano funzionare su un'ampia gamma di piattaforme e ambienti, sono necessari standard per supportare l'interoperabilità. Su questo tipo di standard si è focalizzato il gruppo di lavoro di IETF sulla rilevazione delle intrusioni. Lo scopo del gruppo di lavoro è di definire il formato dei dati e le procedure di scambio dei file per condividere le informazioni di interesse

per i sistemi di rilevazione e risposta alle intrusioni, e per i sistemi di gestione che potrebbero aver necessità di interagire con essi. Ciò che il gruppo di lavoro ha prodotto comprende i seguenti documenti.

1. Un documento dei requisiti che contiene due tipi di requisiti funzionali di alto livello e la loro giustificazione logica. Il primo tipo considera i requisiti per la comunicazione tra i sistemi di rilevazione delle intrusioni, mentre il secondo tipo concerne i requisiti per la comunicazione tra i sistemi di rilevazione delle intrusioni e i sistemi di gestione. Per illustrare i requisiti saranno utilizzati alcuni scenari.
2. Una specifica del linguaggio comune per le intrusioni, che descrive il formato dei dati che soddisfano i requisiti.
3. Un documento di infrastruttura, che identifica quali dei protocolli esistenti è meglio usare per la comunicazione tra i sistemi di rilevazione delle intrusioni e descrive come il formato dei dati ideato si relaziona con questi.

Al momento della scrittura di questo libro, tutti questi documenti sono nello stadio di Internet-draft.

9.3 Gestione delle Password

Protezione delle password

La prima barriera di protezione contro le intrusioni è rappresentata dal sistema delle password. In pratica, tutti i sistemi multiutente richiedono che ciascun utente fornisca non solo un nome o identificatore (ID) ma anche una password. Questa serve ad autenticare l'ID dell'individuo che si sta collegando al sistema. A sua volta, l'ID fornisce sicurezza per i seguenti motivi.

- L'ID determina se l'utente è autorizzato ad avere l'accesso al sistema. In alcuni sistemi, solo gli utenti che possiedono già un ID memorizzato nel sistema sono abilitati all'accesso.
- L'ID determina i privilegi accordati all'utente. Solo un numero ristretto di utenti può possedere il ruolo di supervisore o "superutente" ed essere abilitato alla lettura di file e all'esecuzione di funzioni che risultano specificamente protette a livello di sistema operativo. Alcuni sistemi possiedono account ospite (*guest*) o anonimo (*anonymous*), che offrono agli utenti privilegi limitati.
- L'ID viene utilizzato nel controllo dell'accesso discrezionale. Ad esempio, un utente può conferire il permesso di leggere i propri file ad altri utenti, elencando i loro ID.

Vulnerabilità delle password

Per comprendere la natura degli attacchi ai sistemi basati su password, si consideri lo schema di gestione delle password diffusamente impiegato nei sistemi UNIX, in cui le password non sono mai memorizzate in chiaro. Lo schema opera secondo la seguente procedura (Figura 9.4a). Ciascun utente seleziona una password che si compone di un numero variabile di

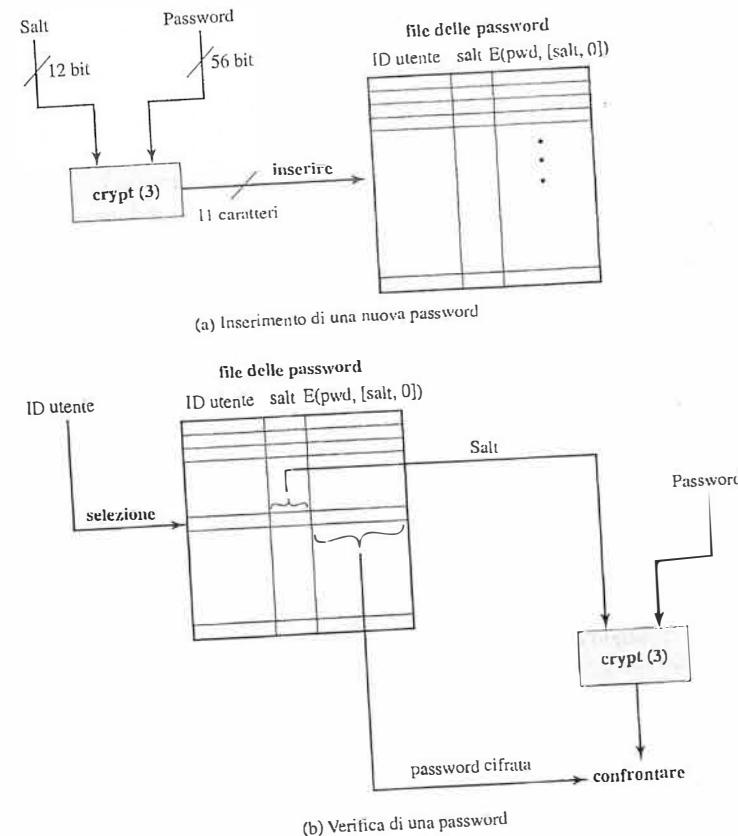


Figura 9.4 Schema di gestione delle password in UNIX.

caratteri stampabili, fino a una lunghezza massima di otto. Questa stringa viene convertita in un valore a 56 bit (utilizzando una rappresentazione ASCII a 7 bit) che costituisce il valore di chiave in ingresso all'algoritmo di cifratura conosciuto come *crypt(3)* e basato su DES. L'algoritmo DES è modificato mediante l'uso di un valore "salt" di 12 bit. In pratica, questo valore è collegato al tempo in cui la password viene assegnata all'utente. L'algoritmo DES così modificato viene eseguito su un ingresso costituito da un blocco di 64 bit di zero. Il risultato prodotto dall'algoritmo viene a sua volta utilizzato come ingresso per un secondo ciclo di cifratura. Questo processo si ripete fino a un totale di 25 cicli di cifratura. Il valore a 64 bit prodotto in uscita è quindi trasformato in una sequenza di 11 caratteri. La password cifrata viene poi memorizzata nel file delle password insieme con la copia in chiaro del valore di salt, in corrispondenza dell'ID dell'utente in questione. Questo metodo si è dimostrato sicuro contro molti attacchi crittoanalitici [WAGN00]. Il valore salt ha tre scopi.

- Evitare password duplicate nel file delle password. Anche se due utenti scegliersero la stessa password, queste sarebbero comunque assegnate in tempi diversi. Pertanto, le due password estese con il corrispondente valore salt differiscono fra loro.
- Aumentare la lunghezza delle password senza richiedere all'utente di dover ricordare due ulteriori caratteri. Il numero di possibili password aumenta quindi di un fattore pari a 4096, rendendo di conseguenza più complessa la scoperta delle password.
- Prevenire l'uso di implementazioni hardware di DES, che semplificherebbero gli attacchi rivolti alla scoperta di password basati sull'approccio a forza bruta.

Quando un utente cerca di connettersi a un sistema UNIX, fornisce l'ID e la password. Il sistema operativo usa l'ID come indice per reperire nel file delle password il valore di salt in chiaro e la password utente cifrata. Il valore di salt e la password fornita dall'utente vengono usati come ingresso dell'algoritmo di cifratura. Se il risultato prodotto dall'algoritmo di cifratura coincide con la password cifrata memorizzata nel file delle password, il sistema accetta la password fornita dall'utente.

L'algoritmo di cifratura è progettato per scoraggiare gli attacchi rivolti alla scoperta di password. Le realizzazioni software di DES sono lente se confrontate con le realizzazioni hardware e l'uso di 25 iterazioni moltiplica di un fattore 25 il tempo necessario all'elaborazione. Tuttavia, sono intervenuti due cambiamenti rispetto alla progettazione originaria dell'algoritmo. In primo luogo le nuove realizzazioni dell'algoritmo hanno guadagnato in velocità di elaborazione. Ad esempio, il noto worm di Internet (descritto nel Capitolo 10) era in grado di scoprire password con qualche centinaio di tentativi, in un tempo ragionevolmente ridotto, utilizzando un algoritmo di cifratura più efficiente rispetto a quello standard del sistema UNIX attaccato. In secondo luogo le prestazioni dell'hardware sono in continua crescita, e quindi qualunque algoritmo software viene eseguito più velocemente.

Due sono le minacce allo schema di gestione delle password in UNIX. Prima di tutto, un utente può ottenere l'accesso a una macchina sfruttando un account ospite o attraverso qualche altra modalità e, successivamente, eseguire su quella macchina un programma di individuazione di password, denominato *password cracker*. L'avversario deve essere in grado di verificare centinaia e forse migliaia di possibili password, consumando risorse di elaborazione ridotte. Inoltre, se un avversario fosse in grado di entrare in possesso di una copia del file delle password, il password cracker potrebbe essere eseguito con comodo su un'altra macchina. Questo consente all'avversario di tentare parecchie migliaia di password in un periodo di tempo ragionevole.

Un esempio di password cracker venne segnalato su Internet nell'agosto del 1993 [MADS93]. Sfruttando un elaboratore parallelo della Thinking Machines Corporation, si ottennero prestazioni di 1.560 operazioni di cifratura al secondo per unità vettore. Con una configurazione standard di quattro unità vettore per nodo di elaborazione, ciò significa 800.000 operazioni di cifratura al secondo su una macchina con 128 nodi (che rappresenta una dimensione modesta), oppure 6,4 milioni di operazioni di cifratura al secondo su una macchina con 1.024 nodi.

Anche con queste significative prestazioni, non è proponibile per un avversario utilizzare una banale tecnica basata sull'approccio "a forza bruta", provando tutte le possibili combinazioni di caratteri al fine di scoprire password valide. Piuttosto, i password cracker sfruttano il fatto che spesso gli utenti utilizzano password facilmente indovinabili.

Lunghezza	Numero	Percentuale sul totale
1	55	0,004
2	87	0,006
3	212	0,02
4	449	0,03
5	1260	0,09
6	3035	0,22
7	2917	0,21
8	5772	0,42
Totale	13787	1,0

Tabella 9.3 Lunghezze di password rilevate [SPAF92a].

In genere gli utenti tendono a scegliere password incredibilmente corte. Nella Tabella 9.3 sono riportati i risultati di uno studio condotto presso la Purdue University. Lo studio era rivolto a osservare le scelte relative alle operazioni di cambiamento di password su 54 macchine, per un totale di circa 7.000 account utente. Dallo studio è emerso che circa il 3 per cento delle password era costituito da tre caratteri o da un numero anche inferiore. Di conseguenza, un avversario potrebbe iniziare il proprio attacco provando esaustivamente tutte le possibili password composte da 3 o da un numero inferiore di caratteri. Un semplice rimedio consiste nel fare in modo che il sistema rifiuti qualunque password costituita da meno di sei caratteri oppure imponga una precisa lunghezza di password, ad esempio otto caratteri.

La lunghezza delle password rappresenta solo una parte del problema. Molte persone, al momento della scelta della propria password, optano per una password che risulta facile da indovinare come, ad esempio, il proprio nome, il proprio indirizzo, un termine comune del vocabolario, e così via. Queste abitudini semplificano il compito del password cracking. L'avversario deve semplicemente controllare il file delle password rispetto a una lista di password probabili. Dato che molti utenti fanno uso di password facili da indovinare, questa strategia dovrebbe idealmente funzionare su qualunque sistema.

[KLE190] riporta una dimostrazione dell'efficacia del processo di individuazione di password. L'autore ha raccolto file di password UNIX provenienti da una vasta gamma di sorgenti per un totale di circa 14.000 password cifrate. Il risultato, definito "preoccupante" dall'autore, è mostrato nella Tabella 9.4. In sintesi, circa un quarto delle password sono state indovinate utilizzando la strategia qui di seguito riportata.

1. Provare con il nome dell'utente, le iniziali, il nome dell'account ed altre informazioni significative di carattere personale. Si tentarono 130 diverse permutazioni in tutto per ciascun utente.
2. Provare con parole prese da vari dizionari. L'autore compilò un dizionario contenente oltre 60.000 vocaboli, includendo il dizionario in linea del sistema e diverse altre liste di parole.

Tipo di password	Dimensione dello spazio di ricerca	Numero di corrispondenze	Percentuale di password combacianti	Percentuale costi/benefici*
Nome di utente/account	130	368	2,7%	2,830
Sequenze di caratteri	866	22	0,2%	0,025
Numeri	427	9	0,1%	0,021
Cinese	392	56	0,4%	0,143
Nomi geografici	628	82	0,6%	0,131
Nomi comuni	2239	548	4,0%	0,245
Nomi femminili	4280	161	1,2%	0,038
Nomi maschili	2866	140	1,0%	0,049
Nomi fuori dal comune	4955	130	0,9%	0,026
Miti e leggende	1246	66	0,5%	0,053
Nomi shakespeariani	473	11	0,1%	0,023
Vocaboli sportivi	238	32	0,2%	0,134
Fantascienza	691	59	0,4%	0,085
Film e attori	99	12	0,1%	0,121
Cartoni animati	92	9	0,1%	0,098
Personaggi famosi	290	55	0,4%	0,190
Frasi e pattern	933	253	1,8%	0,271
Cognomi	33	9	0,1%	0,273
Biologia	58	1	0,0%	0,017
Dizionario di sistema	19683	1027	7,4%	0,052
Nomi di macchine	909	132	1,0%	0,015
Nomi mnemonici	14	2	0,0%	0,143
Bibbia di Re Giacomo	7525	83	0,6%	0,011
Miscellanea di parole	3212	54	0,4%	0,017
Parole Yiddish	56	0	0,0%	0,000
Asteroidi	2407	19	0,1%	0,007
TOTALE	62727	3340	24,2%	0,053

* Calcolato come il numero di corrispondenze diviso per la dimensione dello spazio di ricerca. Tanto maggiore è il numero di parole provate per stabilire una corrispondenza, tanto minore è la percentuale costi/benefici.

Tabella 9.4 Tentativi di scoperta di password su un campione di 13797 account [KLEI90].

3. Provare con varie permutazioni delle parole utilizzate al punto precedente. Questi tentativi comprendevano la sostituzione della prima lettera con la corrispondente maiuscola o con un carattere di controllo, trasformare in lettere maiuscole l'intera parola, usare la parola scritta all'inverso, il cambiamento della lettera "o" con la cifra "zero" e così via. Per effetto di queste permutazioni, venne aggiunto un ulteriore milione di parole alla lista.
4. Provare con varie permutazioni sostituendo lettere maiuscole alle parole al punto 2, non considerate al punto 3. Questo produsse quasi 2 milioni di ulteriori parole che si aggiunsero alla lista.

Di conseguenza, i tentativi si svolsero su un insieme di circa 3 milioni di parole. Utilizzando la più veloce delle configurazioni della Thinking Machines precedentemente descritta, il tempo necessario per eseguire la cifratura di tutte queste parole con tutti i possibili valori salt è inferiore a un'ora.

Controllo dell'accesso

Una modalità per contrastare attacchi basati su password consiste nel negare all'avversario l'accesso al file delle password. Se la porzione cifrata del file delle password è accessibile solamente a un utente privilegiato, allora l'avversario non può leggere il file senza essere a conoscenza della password di un utente privilegiato. [SPAF92a] evidenzia parecchi punti deboli in questa strategia.

- Molti sistemi, fra cui la maggior parte dei sistemi UNIX, sono suscettibili di penetrazioni non previste. Una volta che l'avversario ha in un qualche modo ottenuto l'accesso al sistema, cercherà di ottenere un insieme di password al fine di utilizzare differenti account in sessioni diverse, riducendo così il rischio di essere scoperto. Oppure, un utente con un dato account potrebbe volersi collegare da un altro account per accedere a dati privilegiati o per effettuare sabotaggi al sistema.
- Un incidente nel sistema di protezione potrebbe rendere leggibile il file delle password, compromettendo di conseguenza tutti gli account.
- Alcuni utenti possiedono account su altre macchine in altri domini di protezione, ma utilizzano la stessa password per tutti gli account. Pertanto, se una password viene scoperta da qualcuno su una data macchina, potrebbe risultare compromessa anche una macchina localizzata altrove.

Di conseguenza, risulta maggiormente efficace una strategia in cui gli utenti sono costretti a scegliere password difficili da indovinare.

Strategie di selezione delle password

L'insegnamento, derivato dai due esperimenti appena descritti (Tabelle 9.3 e 9.4), evidenzia come molti utenti, lasciati alla propria iniziativa, scelgano una password troppo corta o troppo semplice da indovinare. All'estremo opposto, se agli utenti venissero assegnate password composte da otto caratteri stampabili assolutamente casuali, ricostruire le password diventerebbe effettivamente impossibile; ma risulterebbe altrettanto impossibile per gli utenti ricordarsi la propria password. Fortunatamente, pur limitando l'universo delle password alle stringhe di caratteri ragionevolmente memorizzabili, lo spazio delle possibili password è ancora troppo ampio per rendere fattibile il processo di password cracking. L'obiettivo, quindi, è quello di eliminare password facili da indovinare e nel contempo consentire all'utente di scegliere una password facile da ricordare. A tale scopo, si utilizzano quattro tecniche:

- educazione degli utenti
- password generate dal calcolatore
- controllo delle password di tipo reattivo
- controllo delle password di tipo proattivo.

Gli utenti dovrebbero essere informati dell'importanza di utilizzare password difficili da scoprire e si potrebbe fornire loro un insieme di linee guida per la scelta di password solide. È improbabile che questa strategia di **educazione degli utenti** abbia successo, specialmente nei sistemi con un elevato numero di utenti o con un frequente ricambio di personale. Infatti, molti ignoreranno semplicemente le linee guida, altri potrebbero non valutare bene cosa sia veramente una password solida. Ad esempio, molti utenti ritengono erroneamente che sia sufficiente invertire una parola oppure sostituire l'ultimo carattere con il corrispondente in maiuscolo per rendere la password non scopribile.

Anche le **password generate dal calcolatore** presentano problemi: se sono intrinsecamente casuali, gli utenti non riescono a ricordarle; ma anche se sono pronunciabili potrebbero risultare di difficile memorizzazione, per cui gli utenti sarebbero tentati di annotarle, scrivendole da qualche parte. In generale, l'esperienza ha mostrato che il livello di accettazione degli utenti per gli schemi di generazione di password è piuttosto basso. Il documento FIPS PUB 181 descrive uno degli schemi di generazione automatica di password meglio progettati. Lo standard contiene sia una descrizione dell'approccio sia tutto il codice sorgente C dell'algoritmo. L'algoritmo genera parole creando sillabe pronunciabili e concatenandole fra loro per formare una parola. Un generatore casuale di numeri produce un flusso casuale di caratteri utilizzato per costruire le sillabe e le parole.

Nella strategia di **controllo delle password di tipo reattivo** il sistema periodicamente esegue il proprio algoritmo di password cracking per trovare password indovinabili. Il sistema cancella qualunque password scoperta, notificandolo all'utente. Questo modo di procedere ha alcuni inconvenienti. Prima di tutto, se il compito viene eseguito correttamente richiede molte risorse di elaborazione. Dato che un avversario determinato, in grado di catturare il file delle password, può dedicare tutto il tempo di CPU a tale compito per ore o anche giorni, un controllo reattivo delle password efficace è un chiaro svantaggio. Inoltre, le password indovinabili che esistono restano vulnerabili fino a quando non sono rilevate dal controllo reattivo.

L'approccio più promettente per migliorare la sicurezza delle password è il **controllo delle password di tipo proattivo**. In questo schema, si lascia facoltà all'utente di scegliere la propria password. Tuttavia, al momento della scelta, il sistema effettua alcuni controlli per verificare che la password sia ammissibile, altrimenti la rifiuta. Tali controlli si basano sull'assunto che, sotto la guida del sistema, gli utenti siano in grado di scegliere password facili da ricordare nell'ambito di uno spazio sufficientemente ampio di possibili password che avranno bassa probabilità di essere scoperte mediante un attacco basato sull'uso di un dizionario.

Il trucco con questa strategia consiste nel trovare un equilibrio fra accettabilità da parte degli utenti e robustezza. Se il sistema rifiutasse troppe password, l'utente potrebbe lamentarsi della difficoltà di scelta. Se il sistema utilizzasse un algoritmo troppo semplice per definire la tipologia di password accettabili, ciò potrebbe fornire informazioni ai cracker per raffinare le loro tecniche deduttive. Nella parte restante di questo paragrafo, si prenderanno in considerazione possibili approcci al controllo proattivo di password.

Il primo approccio è un semplice sistema di applicazione di regole come, ad esempio, le seguenti:

- tutte le password devono avere una lunghezza di almeno otto caratteri;
- la password deve contenere nei primi otto caratteri almeno un carattere maiuscolo, un carattere minuscolo, una cifra e un segno di punteggiatura.

Queste regole potrebbero essere combinate con consigli per l'utente. Anche se tale approccio risulta migliore della semplice educazione degli utenti, potrebbe non essere sufficiente a contrastare gli attacchi dei cracker delle password. Questo schema mette in guardia i cracker su quali password non provare ma rende comunque possibile la scoperta di password.

Un altro approccio adottabile consiste nel compilare un grande dizionario di potenziali password da evitare. Quando un utente sceglie una password, il sistema verifica che non sia presente in tale dizionario. Questo approccio presenta due problemi.

- **Spazio:** per poter essere efficace, il dizionario deve essere molto grande. Ad esempio, il dizionario utilizzato nello studio della Purdue [SPAF92a] occupa più di 30 megabyte di spazio in memoria.
- **Tempo:** il tempo necessario ad effettuare la ricerca in un dizionario di dimensioni elevate può diventare anch'esso elevato. Inoltre, per controllare probabili permutazioni di parole, occorre che queste siano tutte inserite nel dizionario – rendendolo veramente enorme – altrimenti ciascuna ricerca richiederà considerevoli tempi di elaborazione.

Vi sono due promettenti tecniche per sviluppare controlli efficaci ed efficienti delle password di tipo proattivo basate sul rifiuto di parole contenute in una lista. Una di queste tecniche sviluppa un modello di Markov per la generazione di password indovinabili [DAVI93]. La Figura 9.5 illustra una versione semplificata di questo modello basata su un linguaggio con un alfabeto di tre caratteri. Lo stato del sistema in un istante qualunque è il valore della lettera più recente. Il valore posto su una transizione tra due stati rappresenta la probabilità che una lettera ne seguì un'altra. Pertanto la probabilità che la lettera b segua la lettera a è pari a 0,5.

In generale, un modello di Markov è una quadrupla $[m, A, T, k]$, dove m è il numero di stati nel modello, A è lo spazio degli stati, T è la matrice delle probabilità di transizione e k è l'ordine del modello. Per un modello di ordine k , la probabilità che si verifichi una transizione a una particolare lettera dipende dalle k lettere precedenti che sono state generate. La Figura 9.5 mostra un semplice modello del primo ordine.

Gli autori trattano anche lo sviluppo e l'utilizzo di un modello del secondo ordine. Per iniziare, si costruisce un dizionario di password indovinabili. Quindi viene calcolata la matrice delle transizioni.

1. Determinare la matrice delle frequenze f , dove $f(i, j, k)$ è il numero di occorrenze di gruppi di tre caratteri corrispondenti al i -esimo, j -esimo e k -esimo carattere. Ad esempio, la password *parsnips* genera i gruppi di tre caratteri *par*, *ars*, *rsn*, *sni*, *nip*, e *ips*.
2. Per ciascun gruppo di due caratteri ij , calcolare $f(i, j, \infty)$ come il numero totale di gruppi di tre caratteri che iniziano con ij . Ad esempio, $f(a, b, \infty)$ sarebbe il numero totale di gruppi di tre caratteri della forma *aba*, *abb*, *abc* e così via.
3. Calcolare le entrate di T come segue:

$$T(i, j, k) = \frac{f(i, j, k)}{f(i, j, \infty)}$$

Il risultato è un modello che riflette la struttura delle parole nel dizionario. Con questo modello, la domanda "Questa password è facilmente indovinabile?" viene trasformata nella do-

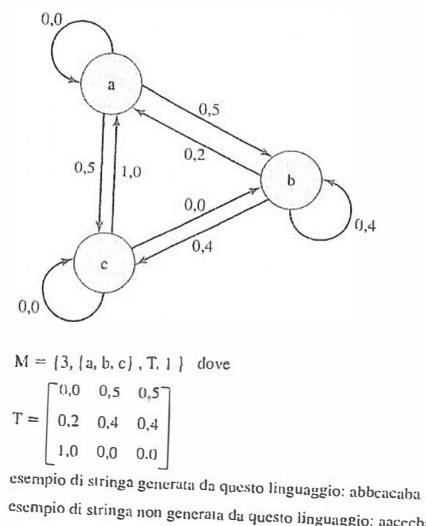


Figura 9.5 Esempio di modello di Markov.

manda "Questa stringa di caratteri (password) è stata generata da questo modello di Markov?". Data una password, si possono osservare le probabilità di transizione di tutti i suoi gruppi di tre caratteri. Si possono usare alcuni test statistici standard per stabilire se la password è plausibile o meno rispetto a un certo modello. Le password che plausibilmente sono generate dal modello sono rifiutate. Gli autori riportano buoni risultati con un modello del secondo ordine. Il loro sistema cattura praticamente tutte le password nel loro dizionario e non esclude così tante password accettabili da risultare poco amichevole per l'utente.

Un differente approccio è quello riportato da Spafford [SPAF92a, SPAF92b]. Tale approccio si basa sull'uso di un filtro di Bloom [BLOO70]. Un filtro di Bloom di ordine k consiste in un insieme di k funzioni hash indipendenti $H_1(x)$, $H_2(x)$, ..., $H_k(x)$, dove ciascuna funzione mette in corrispondenza una password con un valore hash nell'intervallo da 0 a $N - 1$, ovvero:

$$H_i(X_j) = y \quad 1 \leq i \leq k; \quad 1 \leq j \leq D; \quad 0 \leq y \leq N - 1$$

dove:

X_j = j -esima parola nel dizionario delle password

D = numero di parole nel dizionario delle password

Successivamente, si applica al dizionario la procedura che segue.

1. Viene definita una tabella hash di N bit, i cui bit sono tutti posti inizialmente a 0.
2. Per ogni password, si calcolano i suoi k valori hash, ponendo a 1 i corrispondenti bit nella tabella hash. Quindi, se $H_i(X_j) = 67$ per qualche (i, j) , allora si pone a 1 il 67-esimo bit dalla tabella hash; se il bit possiede già il valore 1, il valore non viene modificato.

Data una nuova password, l'algoritmo calcola i suoi k valori hash. Se tutti i bit corrispondenti nella tabella hash sono uguali a 1, allora la password viene rifiutata. Tutte le password del dizionario sono rifiutate. Ci sono anche alcune password cosiddette "false positive" (ovvero password che non sono nel dizionario ma che hanno i corrispondenti bit a 1 nella tabella hash). Per chiarire meglio il concetto, si consideri uno schema con due funzioni hash. Si supponga ora che le password *undertaker* e *bulkhogan* siano contenute nel dizionario, e che *xG%#jj98* non lo sia. Si supponga inoltre che:

$$\begin{array}{lll} H_1(\text{undertaker}) = 25 & H_1(\text{bulkhogan}) = 83 & H_1(\text{xG%#jj98}) = 665 \\ H_2(\text{undertaker}) = 998 & H_2(\text{bulkhogan}) = 665 & H_2(\text{xG%#jj98}) = 998 \end{array}$$

Presentando al sistema la password *xG%#jj98*, verrà rifiutata pur non appartenendo al dizionario. In presenza di troppi falsi positivi, diventa difficile per gli utenti scegliere le password. Pertanto sarebbe opportuno progettare lo schema hash con l'obiettivo di minimizzare il fenomeno dei falsi positivi. Si può mostrare che la probabilità di un falso positivo può essere approssimata da:

$$P \approx (1 - e^{kD/N})^k = (1 - e^{k/R})^k$$

o, in maniera equivalente,

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

dove:

k = numero di funzioni hash

N = numero di bit nella tabella hash

D = numero di parole nel dizionario

$R = N/D$, rapporto fra la dimensione della tabella hash (in bit) e la dimensione del dizionario (in parole)

La Figura 9.6 traccia P in funzione di R per diversi valori di k . Si supponga di avere un dizionario di 1 milione di parole e di voler avere una probabilità 0,01 di rifiutare una password non presente nel dizionario. Scegliendo sei funzioni hash, il rapporto richiesto è $R = 9,6$. Perciò è necessaria una tabella hash di $9,6 \times 10^6$ bit, pari a circa 1,2 Mbyte di memoria. Per contro, la memorizzazione dell'intero dizionario richiederebbe circa 8 Mbyte. Si ottiene pertanto una compressione al più di un fattore 7. Inoltre, la verifica delle password implica il semplice calcolo di sei funzioni hash ed è indipendente dalla dimensione del dizionario, mentre utilizzando l'intero dizionario si ha un notevole attivita di ricerca².

² Sia il modello markoviano che il filtro di Bloom comportano l'utilizzo di tecniche probabilistiche. Nel caso del modello markoviano, esiste una piccola probabilità che alcune password contenute nel dizionario non vengano catturate e altre non comprese nel dizionario siano rifiutate. Nel caso del filtro di Bloom, vi è una piccola probabilità che alcune password non comprese nel dizionario siano rifiutate. Di nuovo si vede come il considerare un approccio probabilistico semplifichi la soluzione (ad esempio, si veda la Nota 1 a piè di pagina nel Capitolo 5).

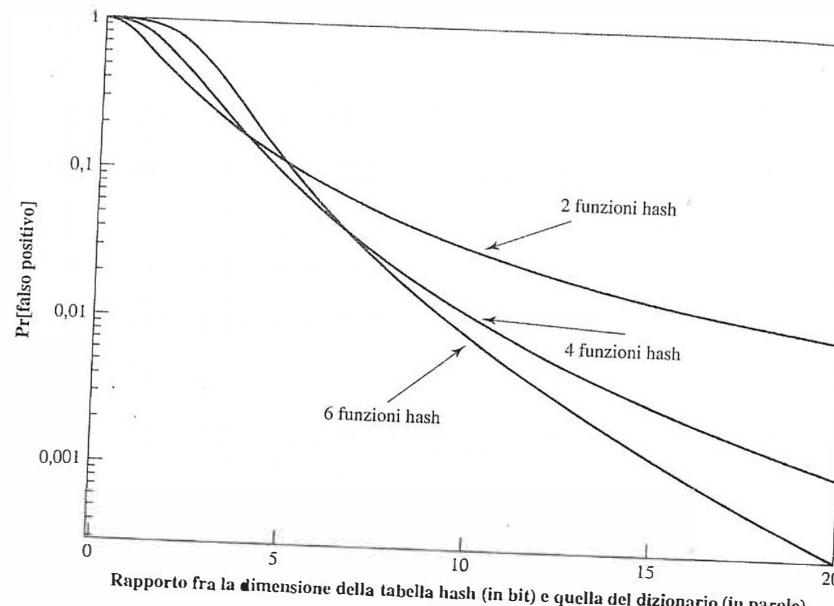


Figura 9.6 Prestazioni del filtro di Bloom.

9.4 Letture consigliate e siti web

[BACE00] e [PROC01] sono due trattazioni approfondite della rilevazione delle intrusioni; [BACE01] è un'esposizione più concisa ma la cui lettura si rivela sicuramente utile. [KENT00] e [MCHU00] sono due brevi ma utili articoli che offrono una rassegna dell'argomento. [NING04] studia i recenti progressi nelle tecniche di rilevazione delle intrusioni. [HONE01] è l'ultimo resoconto sui honeypot e fornisce un'analisi dettagliata degli strumenti e dei metodi degli hacker.



Siti web consigliati

- **CERT Coordination Center:** l'organizzazione sviluppatasi dal CERT, costituita dalla Defense Advanced Research Projects Agency. Il sito fornisce interessanti informazioni su minacce e vulnerabilità della sicurezza Internet, e statistiche sugli attacchi.
- **Progetto Honeynet:** un progetto di ricerca che studia le tecniche dei rapaci hacker e sviluppa prodotti honeypot.
- **Honeypots:** una buona collezione di articoli di ricerca e tecnici.
- **Intrusion Detection Working Group:** comprende tutti i documenti creati da questo gruppo.

9.5 Domande di revisione ed esercizi

Domande di revisione

- 9.1 Elencare e definire brevemente le tre categorie di intrusi.
- 9.2 Quali sono le due comuni tecniche usate per proteggere il file delle password?
- 9.3 Quali sono i tre benefici che possono essere forniti da un sistema di rilevazione delle intrusioni?
- 9.4 Qual è la differenza tra l'individuazione di una anomalia statistica e una rilevazione di intrusione basata su regole?
- 9.5 Quali metriche sono utili in una rilevazione di intrusioni basata sul profilo?
- 9.6 Qual è la differenza tra la rilevazione delle anomalie basata su regole e l'identificazione delle penetrazioni basata su regole?
- 9.7 Che cos'è un honeypot?
- 9.8 Che cos'è il valore di salt nel contesto della gestione delle password di UNIX?
- 9.9 Elencare e definire brevemente le quattro tecniche usate per evitare password facili da indovinare.

Esercizi

- 9.1 Un taxi fu coinvolto in un incidente fatale notturno nel quale il colpevole scappò dopo lo scontro. In città lavorano due compagnie di taxi, la Verde e la Blu. È noto che:
 - 85% dei taxi in città sono di Verde e 15% sono di Blu.
 - Un testimone ha identificato il taxi come Blu.
 Il tribunale verificò l'affidabilità del testimone nelle stesse circostanze che vi erano la notte dell'incidente e concluse che il testimone era attendibile nell'identificazione del colore del taxi nell'80% delle volte. Qual è la probabilità che il taxi coinvolto nell'incidente fosse Blu piuttosto che Verde?
- 9.2 Supponete di scegliere le password fra le possibili combinazioni di quattro caratteri dei 26 caratteri dell'alfabeto. Assumete inoltre che un avversario sia in grado di provare password a un tasso di una al secondo.
 - a. Ipotizzando che non vi siano informazioni di ritorno all'avversario fino a quando un tentativo non sia stato concluso, determinate il tempo necessario per individuare la password corretta.
 - b. Ipotizzando che venga inviata all'avversario una notifica di errore ogni volta che digita un carattere errato, determinate il tempo necessario per individuare la password corretta.
- 9.3 Supponete che elementi sorgente di lunghezza k siano trasformati in elementi obiettivo di lunghezza p , secondo qualche modalità uniforme. Se ciascuna cifra può assumere uno fra r possibili valori, allora il numero di elementi sorgente è pari a r^k e il numero di elementi obiettivo è pari a r^p (che è un numero più piccolo). Uno specifico elemento sorgente x_i viene trasformato in un particolare elemento obiettivo y_j .
 - a. Qual è la probabilità per un avversario di selezionare l'elemento sorgente corretto in un unico tentativo?

- b. Qual è la probabilità che un avversario possa produrre un elemento sorgente diverso x_k ($x_i \neq x_k$) che venga trasformato nello stesso elemento obiettivo y_j ?
- c. Qual è la probabilità per un avversario di produrre l'elemento obiettivo corretto in un unico tentativo?
- 9.4 Un generatore sonetico di password sceglie casualmente due segmenti per ciascuna password di sei caratteri. Il formato di ciascun segmento è CVC (consonante, vocale, consonante), dove $V = \{a, e, i, o, u\}$ e $C = \bar{V}$.
- Qual è lo spazio delle possibili password?
 - Qual è la probabilità che un avversario indovini una password correttamente?
- 9.5 Supponete che le password possano utilizzare solo i 95 caratteri ASCII stampabili e che tutte le password abbiano lunghezza di 10 caratteri. Assumete inoltre di avere un algoritmo di password cracking con un tasso di cifratura di 6,4 milioni di cifrature al secondo. Quanto tempo impiega questo programma per provare esaustivamente tutte le possibili password su un sistema UNIX?
- 9.6 Per i noti rischi del sistema di gestione delle password di UNIX, la documentazione SunOS-4.0 raccomanda di rimuovere il file delle password e di sostituirlo con un file leggibile a tutti gli utenti denominato /etc/publickey. Una voce di questo file, relativa a un certo utente A, contiene l'identificatore dell'utente, ID_A , la sua chiave pubblica, PU_A , e la sua corrispondente chiave privata PR_A . La chiave privata è cifrata utilizzando DES mediante una chiave derivata dalla password P_a fornita dall'utente in fase di connessione. Quando A si collega al sistema, questo decifra $E[PR_A]$ per ottenere KR_A .
 - Il sistema verifica poi che P_a sia stata fornita correttamente. Come?
 - Come può un avversario attaccare tale sistema?
- 9.7 Lo schema di cifratura utilizzato per le password in UNIX è di tipo unidirezionale; non è cioè possibile effettuare il suo inverso. Pertanto, sarebbe più preciso affermare che si tratta effettivamente di un codice hash piuttosto che di una cifratura della password?
- 9.8 Si è detto che l'inclusione del valore salt nello schema di gestione delle password in UNIX aumenta la difficoltà di indovinare password valide di un fattore pari a 4096. Ma il valore salt è memorizzato in chiaro nella stessa entrata della corrispondente password cifrata. Pertanto, i due caratteri del valore salt sono noti all'avversario e non necessitano di essere indovinati. Perché si afferma che il salt aumenta la sicurezza?
- 9.9 Supponete di avere risposto correttamente all'esercizio precedente e di aver compreso il significato del valore salt; ora si pone un altro problema. È possibile contrastare completamente tutti i programmi di password cracking aumentando drasticamente la dimensione del salt, diciamo a 24 o 48 bit?
- 9.10 Si consideri il filtro di Bloom descritto nel Paragrafo 9.3. Sia: k = numero di funzioni hash, N = numero di bit nella tabella hash e D = numero di parole nel dizionario.
- Mostrate che il numero di bit previsto nella tabella hash con valore zero è dato da:

$$\phi = \left(1 - \frac{k}{N}\right)^D$$

- b. Mostrate che la probabilità che una parola in ingresso, non presente nel dizionario venga falsamente accettata come se fosse presente nel dizionario è data da:

$$P = (1 - \phi)^k$$

c. Mostrate che l'espressione del punto precedente può essere approssimata come:

$$P \approx (1 - e^{-kD/N})^k$$

- 9.11 Progettate un sistema di accesso ai file che consenta a certi utenti un accesso ai file in lettura e scrittura a seconda dell'inizializzazione delle autorizzazioni da parte del sistema. Le istruzioni dovrebbero essere nel formato

Lettura (F, Utente A): tentativo dell'utente A di leggere il file F

Scrittura (F, Utente A): tentativo dell'utente A di memorizzare una copia di F, forse modificata.

Ciascun file ha un record dell'intestazione che contiene i privilegi di autorizzazione, cioè una lista di utenti che possono leggerlo o scriverlo. Il file è da cifrare con una chiave che non è condivisa dagli utenti, ma nota solo al sistema.

Appendice 9A La Base-Rate Fallacy

In questa appendice si dimostrerà la base-rate fallacy, iniziando con un ripasso dei più importanti risultati dalla teoria della probabilità.

Probabilità condizionale e indipendenza

Sovente si vuole conoscere una probabilità condizionata da qualche evento. L'effetto della condizione è quello di eliminare alcuni dei risultati dalla spazio campionario. Ad esempio, qual è la probabilità di ottenere una somma pari a 8 ogni volta che si lancia una coppia di dadi se sappiamo che la faccia di uno dei dadi mostrerà un numero pari? Si può ragionare come segue. Siccome un dado ha fornito un risultato pari, il secondo dado deve mostrare un numero pari; quindi ci sono 36 - (numero di tiri nei quali entrambe le facce dei dadi sono dispari) = 36 - 3 × 3 = 27. La probabilità risultante è $3/27 = 1/9$.

Formalmente, la probabilità condizionale di un evento A, assumendo che si è verificato un evento B, indicata con $\Pr[A|B]$, si definisce come il rapporto:

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]}$$

Dove si è assunto che $\Pr[B]$ non è zero.

Nel nostro esempio, $A = \{\text{somma pari a } 8\}$ e $B = \{\text{almeno un dado pari}\}$. La quantità $\Pr[AB]$ comprende tutti quei risultati nei quali la somma è pari a 8 e almeno uno dei dadi è pari. Come si può vedere, ci sono 3 di questi risultati. Quindi $\Pr[AB] = 3/36 = 1/12$. Pensandoci, ci si dovrebbe convincere che $\Pr[B] = 3/4$. Si può calcolare ora:

$$\Pr[A|B] = \frac{1/12}{3/4} = \frac{1}{9}$$

Questo concorda con i precedenti ragionamenti.

Due eventi A e B sono detti indipendenti se $\Pr[AB] = \Pr[A]\Pr[B]$. Si può facilmente vedere che se A e B sono indipendenti, $\Pr[A|B] = \Pr[A]$ e $\Pr[B|A] = \Pr[B]$.

Teorema di Bayes

Uno dei più importanti risultati della teoria delle probabilità è noto come Teorema di Bayes. In primo luogo è necessario spiegare la formula della probabilità totale. Dato un insieme di eventi mutuamente esclusivi E_1, E_2, \dots, E_n , tali che l'unione di questi eventi includa tutti i possibili risultati, e dato un evento arbitrario A , allora si può mostrare che

$$\Pr[A] = \sum_{i=1}^n \Pr[A|E_i] \Pr[E_i] \quad (9.1)$$

Il Teorema di Bayes può essere espresso come segue:

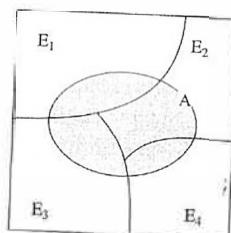
$$\Pr[E_i|A] = \frac{\Pr[A|E_i] \Pr[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i] \Pr[E_i]}{\sum_{i=1}^n \Pr[A|E_i] \Pr[E_i]} \quad (9.2)$$

La Figura 9.7a illustra i concetti della probabilità totale e del Teorema di Bayes.

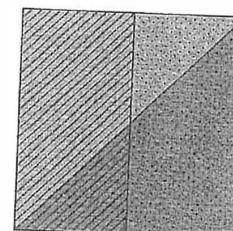
Il Teorema di Bayes viene usato per calcolare la probabilità a posteriori, cioè la probabilità che qualcosa è realmente in questo modo, avendo fatti che lo testimoniano. Ad esempio, ipotizziamo che si stia trasmettendo una sequenza di 0 e di 1 su una linea di trasmissione rumorosa. Siano S_0 e S_1 gli eventi "invio di uno zero in un certo istante" e "invio di un uno in un certo istante" rispettivamente, e R_0 e R_1 siano gli eventi di "ricezione di uno zero" e "ricezione di un uno".

Si ipotizzi di conoscere la probabilità della sorgente, cioè $\Pr[S_1] = p$ e $\Pr[S_0] = 1-p$. Ora si oservi la linea per determinare quanto frequentemente si verifichi un errore quando viene inviato un uno e quando viene inviato uno zero; vengono così calcolate le seguenti probabilità: $\Pr[R_0|S_1] = p_a$ e $\Pr[R_1|S_0] = p_b$. Se viene ricevuto zero, si può calcolare la probabilità condizionale di un errore, cioè la probabilità condizionale che è stato inviato un uno, dato che è stato ricevuto uno zero, usando il Teorema di Bayes:

$$\Pr[R_1|S_0] = \frac{\Pr[S_0|S_1] \Pr[S_1]}{\Pr[R_0|S_1] \Pr[S_1] + \Pr[R_1|S_0] \Pr[S_0]} = \frac{p_a p}{p_a p + (1 - p_b)(1 - p)}$$



(a) Diagramma per illustrare i concetti



(b) Esempio

Figura 9.7 Rappresentazione della probabilità totale e del Teorema di Bayes.

Nella Figura 9.7b è illustrata la precedente equazione. Nella figura lo spazio dei campioni è rappresentato da un quadrato unitario. La metà del quadrato corrisponde a S_0 e l'altra metà a S_1 , così che $\Pr[S_0] = \Pr[S_1] = 0,5$. Analogamente, la metà del quadrato corrisponde a R_0 e l'altra metà a R_1 , così che $\Pr[R_0] = \Pr[R_1] = 0,5$. All'interno dell'area rappresentata da S_0 , $1/4$ di quell'area corrisponde a R_1 , così che $\Pr[R_1|S_0] = 0,25$. Le altre probabilità condizionali si ricavano in modo analogo.

Dimostrazione della Base-Rate Fallacy

Si consideri la seguente situazione. Un esame diagnostico ha avuto esito positivo (cioè indica che il paziente è affatto da una malattia). Si sa che:

- l'accuratezza dell'esame è dell'87% (cioè, se il paziente ha la malattia, nell'87% delle volte l'esame fornirà il risultato corretto e, se il paziente non ha la malattia, nell'87% delle volte l'esame darà il risultato corretto);
- l'incidenza della malattia nella popolazione è 1%.

Dato che il test è positivo, qual è la probabilità che il paziente non abbia la malattia diagnosticata? Cioè qual è la probabilità che questo risulti un falso allarme? È necessario usare il Teorema di Bayes per ottenere una risposta corretta:

$$\Pr[\text{sano/positivo}] = \frac{\Pr[\text{positivo/sano}] \Pr[\text{sano}]}{\Pr[\text{positivo/malato}] \Pr[\text{malato}] + \Pr[\text{positivo/sano}] \Pr[\text{sano}]} = \\ = \frac{(0,13)(0,99)}{(0,87)(0,01) + (0,13)(0,99)} = 0,937$$

quindi, nella maggioranza dei casi, quando si diagnostica una malattia, si tratta di un falso allarme.

Questo problema, [PIAT91], fu presentato a molte persone. Molte fornirono come risposta il 13%. La maggioranza, compresi molti medici, indicò un numero al di sotto del 50%. Molti tra i medici, che avevano fornito una risposta errata, si lamentarono che – se ciò fosse vero – risulterebbe inutile eseguire esami clinici. La ragione per cui la maggior parte delle persone fornì la risposta sbagliata è che non tennero conto del tasso base di incidenza (il base-rate) quando approcciavano intuitivamente il problema. Questo errore è noto come **Base-Rate Fallacy**.

Come si può risolverlo? Si supponga di portare entrambi i tassi di risultato corretto al 99,9%, cioè si ipotizzi di avere $\Pr[\text{positivo/malato}] = 0,999$ e $\Pr[\text{negativo/sano}] = 0,999$. Mettendo questi numeri nell'Equazione 9.2, si ottiene $\Pr[\text{sano/positivo}] = 0,09$. Quindi, se si può determinare con precisione sia la malattia sia la sua assenza a un livello del 99,9%, allora il tasso di falsi allarmi sarà del 9%. Un risultato più soddisfacente, ma non ancora perfetto. Inoltre, si assuma di nuovo che l'accuracy sia del 99,9%, ma ora si supponga che l'incidenza della malattia nella popolazione sia solo del $1/10000 = 0,0001$. Si perviene, allora, a un tasso di falsi allarmi del 91%. In una situazione reale, [AXEL00] calcolò che la probabilità associata con i sistemi di rilevazione delle intrusioni era tale da fornire un tasso di falsi allarmi troppo elevato.

Capitolo 10

Software dolosi

10.1 Virus e minacce correlate

I programmi che sfruttano le vulnerabilità dei sistemi di calcolo rappresentano forse il tipo più sofisticato di minaccia ai sistemi di elaborazione. In questo contesto, ci occuperemo dei programmi applicativi e dei programmi di utilità, quali compilatori ed editor.

Il paragrafo inizia con una panoramica di questo tipo di minacce. Il prosieguo del paragrafo è dedicato a virus e worm.

Programmi dolosi

La terminologia in questo campo presenta svariati problemi dovuta alla mancanza di un accordo generale sui termini e alla sovrapposizione di alcune categorie. La Tabella 10.1, basata principalmente su [SZOR05], rappresenta un'utile guida.

I software dolosi possono essere suddivisi in due categorie: quelli che hanno bisogno di un programma ospite e quelli indipendenti. I primi sono essenzialmente frammenti di programma che non possono esistere indipendentemente da qualche programma applicativo reale, un'utilità o un programma di sistema. Alcuni esempi sono i virus, le bombe logiche e le backdoor. I secondi sono programmi auto contenuti che possono essere programmati e fatti eseguire dal sistema operativo. I programmi di worm e zombie ne sono un esempio.

Si può inoltre distinguere tra le minacce software che si auto-replicano e quelle che non lo fanno. Le prime possono essere costituite da frammenti di programma o da programmi indipendenti che, quando eseguiti, possono riprodurre una o più copie di se stessi, attivabili successivamente sullo stesso sistema o su un sistema diverso. Virus e worm sono esempi. Le seconde sono rappresentate da programmi o da frammenti di programmi che si attivano quando viene invocato il programma host per l'esecuzione di una specifica funzione. Esempi di questo tipo di software sono le bombe logiche, la backdoor e i programmi zombie.

Nella parte rimanente di questo paragrafo proporremo una breve panoramica delle categorie principali di software doloso, con l'eccezione di virus e worm che saranno trattati in dettaglio più avanti.

Nome	Descrizione
Virus	Si attacca a un programma e propaga copie di se stesso ad altri programmi.
Worm	Programma che propaga copie di se stesso ad altri computer.
Bombe logiche	Iniziano delle azioni quando si verifica una certa condizione
Cavalli di Troia (trojan horse)	Programmi che contengono funzionalità aggiuntive inaspettate.
Backdoor (trapdoor)	Modifiche ai programmi che permettono l'accesso non autorizzato a certe funzionalità.
Exploit	Codice specifico di una singola vulnerabilità o di un insieme di vulnerabilità.
Scaricatori	Programmi che installano altri elementi sulla macchina che è sotto attacco. Di solito uno scaricatore viene inviato in una e-mail.
Auto-rooter	Strumenti nocivi usati dagli hacker per entrare in una nuova macchina da remoto.
Kit (generatore di virus)	Insieme di strumenti per generare automaticamente nuovi virus.
Programme di spam	Usati per mandare un gran numero di mail indesiderate.
Flooders (inondatori)	Usati per attaccare un computer connesso in rete con un grande volume di traffico per eseguire un attacco di negazione del servizio (DoS).
Keyloggers (registratori di chiavi)	Catturano i tasti pigiati in un sistema compromesso.
Rootkit	Insieme di strumenti per gli hacker utilizzati dopo che l'attaccante è entrato in un computer e ha ottenuto i privilegi di amministratore della macchina.
Zombie	Programmi attivati sulla macchina infettata che vengono avviati per lanciare un attacco alle altre macchine.

Tabella 10.1 Azioni del modello USTAT e corrispondenti tipi di eventi SunOS.

Backdoor

Una backdoor, nota anche come trapdoor, è un punto di accesso segreto a un programma, che consente – a chi ne è a conoscenza – di entrare senza attraversare le consuete procedure di controllo dell'accesso. I programmatore hanno legittimamente utilizzato per molti anni le backdoor con il fine di rilevare e correggere gli errori e per testare i programmi. Ciò viene normalmente fatto quando il programmatore sta sviluppando un'applicazione che possiede una procedura di autenticazione, o una lunga configurazione, per cui l'utente deve inserire

molteplici valori diversi per eseguire l'applicazione. Per effettuare il debugging del programma, lo sviluppatore potrebbe voler acquisire privilegi speciali o evitare tutte le procedure di inizializzazione e autenticazione. Inoltre, potrebbe volersi assicurare un sistema alternativo per attivare l'applicazione nell'eventualità che si verificassero malfunzionamenti nella procedura di autenticazione durante la realizzazione del programma. La trapdoor è un codice che riconosce alcune particolari sequenze in ingresso o che si attiva quando viene eseguito da un particolare ID utente o da un'improbabile sequenza di eventi.

Le trapdoor divengono una minaccia quando sono utilizzate da programmatore senza scrupoli, al fine di ottenere accessi non autorizzati. La trapdoor fu l'idea alla base dell'attacco perpetrato nel film *War Games*. Un ulteriore caso è quello verificatosi durante lo sviluppo di Multics, in cui furono condotti test di penetrazione da parte di un "tiger team" (che simulava gli avversari) dell'Air Force. Una tattica prevedeva l'invio di un finto aggiornamento del sistema operativo a un sito dove girava Multics. L'aggiornamento conteneva un cavallo di Troia (descritto più avanti) che poteva essere attivato da una trapdoor e che consentiva al tiger team di ottenere l'accesso. La minaccia fu così ben implementata che gli sviluppatori di Multics non riuscirono a trovarla, anche dopo essere stati informati della sua presenza [ENGE80]. È molto difficile sviluppare controlli per trapdoor a livello di sistema operativo. Le misure di sicurezza devono concentrarsi sulle attività di sviluppo del programma e di aggiornamento del software.

Bomba logica

Una delle prime tipologie di minaccia, precedente a virus e worm, è la bomba logica. La bomba logica è un codice contenuto in qualche programma legittimo impostato per "esplosione" al verificarsi di specifiche condizioni. Esempi di condizioni che possono essere impiegate come innesco (*trigger*) di una bomba logica sono la presenza o l'assenza di specifici file, un particolare giorno della settimana o una data particolare, o l'esecuzione di un'applicazione da parte di uno specifico utente. Una volta innescata, una bomba può produrre l'alterazione o la cancellazione di dati o di interi file, causare blocchi delle macchine o altri danni. Un esempio sorprendente di come le bombe logiche possono essere impiegate è rappresentato dal caso di Tim Lloyd, condannato per aver messo a punto una bomba logica che costò al suo datore di lavoro, Omega Engineering, più d 10 milioni di dollari, facendo fallire le strategie di crescita della sua società e portando al licenziamento di 80 lavoratori [GAUD00]. Alla fine Lloyd fu condannato a 41 mesi di prigione e gli fu ordinato di pagare 2 milioni di dollari come risarcimento.

Cavalli di Troia

Un cavallo di Troia è un programma o una procedura di comando che svolge un compito utile, o apparentemente utile. Al suo interno è però inserito un codice nascosto che, quando richiamato, esegue alcune funzioni indesiderate o dannose.

I cavalli di Troia possono essere impiegati per svolgere indirettamente funzioni che un utente non autorizzato non potrebbe svolgere direttamente. Ad esempio, un utente potrebbe creare un cavallo di Troia per accedere ai file di un altro utente in un sistema condiviso; quando eseguito, questo modifica i permessi di accesso rendendo i file leggibili da chiunque. L'autore del cavallo di Troia potrebbe quindi spingere altri utenti ad eseguire il programma posizionandolo in una directory comune e denominandolo in modo che sembri un programma di utilità comune. Un esempio è costituito da un programma che all'apparenza fornisce la li-

sta dei file utente in un formato desiderato. Dopo che un altro utente ha eseguito il programma, l'autore può avere accesso alle informazioni contenute nei file di quell'utente. Un tipo di programma cavallo di Troia difficile da individuare è un compilatore modificato al fine di consentire l'inserimento di codice aggiuntivo all'interno di specifici programmi in fase di compilazione, come ad esempio programmi di connessione al sistema [THOM84]. Il codice crea una trapdoor nel programma di connessione che consente all'autore di accedere al sistema utilizzando una password speciale. Questo cavallo di Troia non potrà mai essere scoperto mediante la lettura del codice sorgente del programma di connessione.

La distruzione di dati è un'ulteriore finalità per l'impiego di cavalli di Troia. Apparentemente il programma sembra svolgere funzioni utili (come programmi di calcolo), mentre con tranquillità può anche effettuare la cancellazione dei file dell'utente che – ignaro – ha richiesto l'esecuzione del programma. Un caso simile è accaduto a un dirigente della CBS, vittima di un cavallo di Troia che gli cancellò tutte le informazioni contenute nella memoria del suo calcolatore [TIME90]. Il cavallo di Troia era inserito in una procedura grafica, collocata in un BBS.

Zombie

Si tratta di un programma che, segretamente, prende il controllo di un altro computer connesso a Internet e poi lo utilizza per lanciare attacchi che risultano difficili da far risalire al creatore dello zombie. Gli zombie vengono usati negli attacchi di negazione del servizio, tipicamente contro siti web designati come bersaglio. Lo zombie viene messo di nascosto su centinaia di computer appartenenti a terze parti insospettabili e poi viene utilizzato per soffoccare i siti web designati come bersaglio, lanciando un pesante e soffocante attacco con traffico Internet. Nel Paragrafo 10.3 saranno trattati gli zombie nel contesto degli attacchi di negazione del servizio.

Natura dei virus

Un virus è un pezzo di software che può "infettare" altri programmi modificandoli; la modifica comprende la replica del programma virus, il quale può quindi continuare a infettare altri programmi.

I virus biologici sono frammenti molto piccoli di codice genetico – DNA o RNA – che possono prendere il controllo dei meccanismi di una cellula vivente e la convincono con l'inganno a fare migliaia di repliche perfette del virus originale. Come la sua controparte biologica, il virus dei computer porta, nelle proprie istruzioni di codice, la chiave per fare copie perfette di se stesso. Il virus tipico diventa parte integrante di un programma su un computer. Pertanto, ogni volta il computer infettato viene in contatto con un pezzo di software non infettato, una nuova copia del virus passa al nuovo programma. Ne consegue che l'infezione può essere diffusa da un computer all'altro da utenti che, non sospettando nulla, si scambiano dischi o si inviano reciprocamente programmi su una rete. In un ambiente di rete, la capacità di accedere alle applicazioni e ai servizi di sistema su un altro computer fornisce un terreno perfetto per la diffusione di un virus.

Un virus può eseguire qualsiasi cosa facciano gli altri programmi. La sola differenza è che si "attaccano" agli altri programmi e sono eseguiti segretamente, quando il programma ospite è in funzione. Una volta che il virus è in esecuzione, può effettuare qualsiasi opera-

zione, come cancellare file e programmi. Durante il proprio ciclo di vita, un tipico virus passa attraverso quattro fasi.

- **Fase latente:** il virus è inattivo; sarà avviato da qualche evento come una data, la presenza di un altro programma o di un file o il superamento di uno specifico limite di capacità del disco fisso. Non tutti i virus attraversano questa fase.
- **Fase di propagazione:** il virus colloca una copia identica di se stesso in un altro programma o in specifiche aree di sistema del disco. Ogni programma infetto contiene ora un clone del virus, che entrerà a sua volta nella fase di propagazione.
- **Fase di innesto:** il virus viene attivato per eseguire la funzione per cui è stato concepito. Come per la fase latente, l'attivazione del virus può essere causata da una molteplicità di eventi di sistema, incluso il conteggio del numero di volte che la copia corrente del virus si è auto-replicata.
- **Fase di esecuzione:** la funzione per cui il virus è stato concepito viene eseguita. La funzione può essere innocua, come un messaggio sullo schermo, o nociva, come la distruzione di programmi e file di dati.

La maggior parte dei virus svolge la propria azione secondo modalità che dipendono dai particolari sistemi operativi e, in alcuni casi, dalle particolari piattaforme hardware. Di conseguenza, sono progettati per sfruttare dettagli e punti deboli degli specifici sistemi.

Struttura dei virus

Un virus può essere inserito in testa o in coda a un programma eseguibile, oppure può essere allegato seguendo differenti modalità. Il punto cruciale del meccanismo di funzionamento è che il programma infetto, quando lanciato, esegue prima il codice del virus e successivamente il codice originario del programma.

Una descrizione molto generale della struttura del virus è mostrata nella Figura 10.1 [COHE94]. In questo caso, il codice del virus, V, è posto in testa ai programmi infetti e si assume che il punto di ingresso al programma, quando richiamato, sia la prima linea del programma stesso.

Un programma infetto inizia con il codice del virus e funziona nel seguente modo. La prima riga del codice è un salto al programma principale del virus. La seconda riga è uno speciale marcitore utilizzato dal virus per stabilire se un programma vittima potenziale è già stato infettato dal virus. Quando il programma infettato viene richiamato, il controllo passa immediatamente al programma principale del virus. Dapprima il virus ricerca e infetta file eseguibili non ancora infettati. Successivamente, può svolgere qualche azione, in genere dannosa per il sistema. Tale azione può essere svolta ogni volta che il programma viene richiamato, oppure può essere un bomba logica che si attiva solo sotto determinate condizioni. Infine, il virus trasferisce il controllo al programma originario. Se la fase di infezione del programma è sufficientemente rapida, è improbabile che un utente noti differenze tra l'esecuzione di un programma "sano" e quella di uno "infetto".

Un virus come quello appena descritto può essere facilmente individuato, in quanto la versione infettata del programma è più lunga di quella non infettata corrispondente. Un modo per contrastare questo semplice modo di identificazione dei virus è quello di eseguire la com-

```

programma V :=

(goto principale;
1234567;

procedura infezione-esegibile :=
  (loop:
    file := selezione-casuale-file-esegibile;
    if (prima-riga-del-file = 1234567)
      then goto loop
    else anteponi V al file; )

procedura esegui-danneggiamento :=
  (qualsiasi azione dannosa)

procedura verifica-condizione :=
  (ritorna valore vero se si è verificata
  qualche condizione fra quelle previste)

principale: programma-principale :=
  (infezione-esegibile;
  if verifica-condizione then
    esegui-danneggiamento;
  goto prossimo;)

prossimo:
}

```

Figura 10.1 Semplice esempio di virus.

pressione di un file esegibile in modo che le versioni infette e non abbiano identica lunghezza. La Figura 10.2 [COHE94] raffigura in termini generali la logica necessaria a tale scopo. In questo virus le righe di codice rilevanti sono numerate, e la Figura 10.3 [COHE94]

```

programma CV :=

(goto principale;
01234567;

procedura infezione-esegibile :=
  (loop:
    file := selezione-casuale-file-esegibile;
    if (prima-riga-del-file = 01234567) then goto loop
    (1) comprimi il file;
    (2) anteponi CV al file;
  }

principale: programma-principale :=
  (if chiedi-permesso then infezione-esegibile;
  (3) decomprimi resto-del-file;
  (4) esegui il file decompresso; )
}

```

Figura 10.2 Logica di un virus che esegue la compressione dei file.

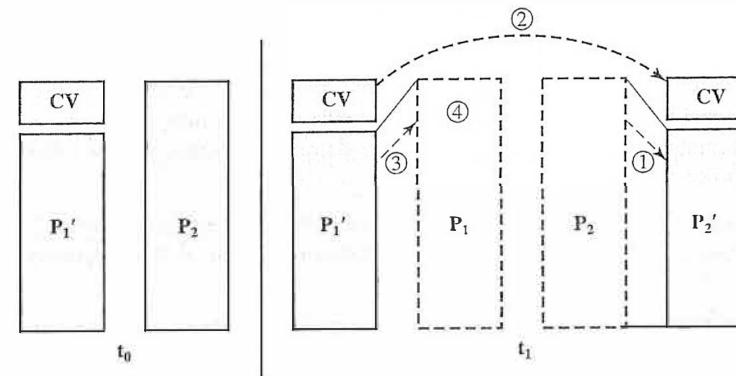


Figura 10.3 Virus con compressione.

ne illustra il funzionamento. Si assuma che il programma P_1 sia infettato dal virus CV. Quando questo programma viene richiamato, il controllo passa al virus ivi contenuto, il quale svolge i seguenti passi.

1. Per ogni file P_2 non infettato che viene individuato, il virus dapprima esegue la compressione del file che produce come risultato un file P'_2 più corto dell'originario di una quantità pari alla dimensione del virus stesso.
2. Viene inserita in testa al programma compresso una copia del virus.
3. La versione compressa del programma originario infetto, P'_1 , viene decompressa.
4. Viene eseguito il programma originario decompresso.

In questo esempio, il virus non fa nient'altro che propagarsi e, come nell'esempio precedente, può contenere una bomba logica.

Infezione iniziale

Una volta che il virus è riuscito a penetrare un sistema infettando un singolo programma, è in condizione di infettare alcuni o tutti gli altri file eseguibili presenti nel sistema, non appena il programma infetto viene eseguito. Pertanto, l'infezione virale può essere prevenuta solo impedendo il primo accesso del virus nel sistema. Sfortunatamente, la prevenzione è molto difficile in quanto un virus può essere parte di qualsiasi altro programma esterno al sistema. Di conseguenza, chiunque è vulnerabile, a meno di voler prendere una macchina "nuda" e scrivere autonomamente tutti i programmi di sistema e le applicazioni.

Tipologie di virus

Fin dall'iniziale comparsa dei virus si è instaurata una battaglia continua fra i progettisti di virus e quelli di software antivirus. Così, mentre venivano sviluppate contromisure appropriate per i virus esistenti, ne venivano creati di nuovi. [STEP93] indica come più significative le seguenti categorie di virus.

- **Virus parassiti:** la tradizionale e ancora più comune tipologia di virus. Un virus parassita inserisce una copia di se stesso all'interno di file eseguibili e si replica quando il programma infetto viene eseguito, trovando altri file eseguibili da infettare.
- **Virus residenti in memoria:** si installano nella memoria principale come parte di un programma di sistema residente. A partire da quel momento, il virus infetta qualunque programma venga eseguito.
- **Virus del settore di boot:** infetta un record di boot principale o un record di boot di un disco e si diffonde nel momento in cui viene eseguito il caricamento del sistema operativo dal disco contenente il virus.
- **Virus furtivo:** tipologia di virus progettata esplicitamente per risultare non rilevabile da parte del software antivirus.
- **Virus polimorfo:** virus mutante che varia a ogni processo di infezione, rendendo impossibile la sua rilevazione sulla base della "signature".
- **Virus Metamorfici:** come i virus polimorfici quelli metamorfici mutano ad ogni infezione. La differenza è che un virus metamorfico si riscrive completamente a ogni infezione, aumentando la difficoltà nella rilevazione. I virus metamorfici possono cambiare il loro comportamento, come pure il loro aspetto.

Un esempio di **virus furtivo** è un virus che utilizza le tecniche di compressione in modo che la lunghezza del programma infetto sia esattamente la stessa del programma originario, prima dell'infezione. Sono possibili anche tecniche più sofisticate. Ad esempio, un virus può inserire una logica di intercettazione all'interno delle procedure di I/O del disco in modo tale che quando c'è un tentativo di leggere porzioni sospette del disco utilizzando queste procedure, il virus mostrerà il programma originario non infetto. Pertanto, l'aggettivo "furtivo" non si applica direttamente al virus in quanto tale, ma piuttosto indica una tecnica utilizzata dal virus per non essere rilevato.

Un **virus polimorfo**, durante la fase di replica, crea copie funzionalmente equivalenti di se stesso, che però possiedono sequenze di bit decisamente diverse. Come nel caso di virus furtivi, lo scopo è quello di sconfiggere i programmi di scansione antivirus. In questo caso, la **signature** del virus cambierà per ciascuna copia. Per ottenere la variazione della signature, il virus può inserire in modo casuale istruzioni superflue oppure scambiare l'ordine di istruzioni indipendenti. Un approccio più efficace si basa sull'utilizzo della cifratura. Una parte del virus, generalmente denominata **motore di mutazione** (*mutation engine*), genera una chiave di cifratura casuale con cui cifrare la parte restante del virus stesso. La chiave è memorizzata insieme al virus e anche il motore di mutazione viene modificato. All'invocazione di un programma infetto, il virus utilizza la chiave casuale memorizzata per eseguire la decifratura di se stesso. In fase di replica, il virus genera una chiave casuale diversa.

Un'ulteriore arma a disposizione degli ideatori di virus è rappresentata dal toolkit per la creazione di virus. Questo permette a un progettista di virus principiante di creare velocemente un certo numero di virus diversi. Anche se i virus creati con l'ausilio di toolkit risultano meno sofisticati rispetto ai virus progettati *ex-novo*, il numero di virus che si possono generare rappresenta un problema per gli schemi antivirus.

Virus delle macro

Nella metà degli anni '80, i virus delle macro divennero di gran lunga il tipo di virus più diffuso. I virus delle macro sono particolarmente pericolosi per molteplici ragioni.

1. Sono indipendenti dalla piattaforma sottostante. Praticamente, tutti i virus delle macro infettano i documenti Microsoft Word e possono quindi infettare tutte le piattaforme hardware e i sistemi operativi che supportano questa applicazione.
2. I virus delle macro infettano documenti e non porzioni di codice eseguibile. La maggior parte delle informazioni è introdotta nel calcolatore sotto forma di documento piuttosto che di programma.
3. Questa tipologia di virus si diffondono rapidamente, in particolare attraverso la posta elettronica.

Questo tipo di virus sfrutta una caratteristica peculiare di Word e di altre tipiche applicazioni di supporto al lavoro di ufficio, come Microsoft Excel: le macro. In sostanza, una macro è un programma eseguibile, contenuto in documenti elettronici o in altri tipi di file, utilizzato per automatizzare compiti di tipo ripetitivo. Il linguaggio delle macro generalmente è una variante del linguaggio di programmazione Basic. Un utente può definire all'interno di una macro una sequenza di digitazioni da tastiera e fare in modo che la macro sia invocata digitando un tasto funzionale oppure una breve combinazione speciale di tasti.

Le più recenti versioni di Word forniscono maggiore protezione dai virus delle macro. Ad esempio, Microsoft offre uno strumento facoltativo, Macro Virus Protection, che individua i file Word sospetti e mette in guardia l'utente sul potenziale rischio connesso all'apertura di file contenenti macro. Anche diversi produttori di antivirus hanno sviluppato strumenti per la rilevazione e la correzione dei virus delle macro. Come per le altre tipologie di virus, anche in questo caso la battaglia con i loro creatori continua, anche se attualmente non costituiscono più la minaccia predominante.

Virus della posta elettronica

Uno sviluppo più recente nel software doloso è il virus della posta elettronica. I primi virus della posta elettronica, che si sono diffusi rapidamente, come Melissa, facevano uso di una macro di Microsoft Word all'interno di un allegato. Se il ricevente apre l'allegato della posta elettronica, la macro di Word si attiva. Ecco cosa avviene.

- Il virus della posta elettronica invia se stesso a quanti sono presenti nella lista dei contatti dell'applicazione di posta elettronica dell'utente.
- Il virus arreca danni localmente.

Alla fine del 1999, apparve una versione più potente del virus della posta elettronica. Questa nuova versione può essere attivata solamente aprendo la mail che contiene il virus, anziché aprire un allegato. Il virus usa un linguaggio di scripting Visual Basic, supportato dall'insieme delle applicazioni di posta elettronica.

Si è testimoni, quindi, di una nuova generazione di software doloso che arriva via posta elettronica e che usa le caratteristiche dei software di posta elettronica per replicarsi attraverso Internet. Il virus si propaga, appena si è attivato (sia aprendo un allegato alla posta

sia apendo la posta stessa), a tutti gli indirizzi di posta elettronica noti all'host infettato. Come risultato, mentre i virus impiegavano mesi o anni a propagarsi, ora lo fanno in poche ore. Questo rende molto difficile per un software antivirus reagire prima che vengano fatti molti danni. Infine, si dovrebbe inserire un maggiore grado di sicurezza all'interno delle utilità di Internet e nelle applicazioni software, per neutralizzare la crescente minaccia.

Worm

Si tratta di un programma che può replicarsi e inviare copie di se stesso da un computer all'altro attraverso le connessioni di rete. Al momento del suo arrivo, può essere attivato per replicarsi e propagarsi ancora. Oltre alla propagazione, un worm di solito esegue qualche operazione indesiderata. Un virus della posta elettronica ha alcune delle caratteristiche di un worm, perché si propaga da un sistema ad un altro. Tuttavia, si può ancora classificarlo come virus in quanto richiede un'azione umana per la sua diffusione. Un worm si adopera attivamente per trovare più macchine da infettare e ciascuna macchina infettata serve da trampolino di lancio automatico per attacchi ad altre macchine.

I programmi worm di rete usano le connessioni di rete per diffondersi da un sistema all'altro. Una volta attivato all'interno di un sistema, un worm di rete può comportarsi come un virus del computer o un batterio, oppure può installare programmi cavalli di Troia, o eseguire un numero qualsiasi di azioni malefiche e distruttive. Ecco alcuni esempi di mezzi che un worm può utilizzare per auto-replicarsi.

- **Sistemi di posta elettronica:** un worm invia una copia di se stesso ad altri sistemi trasmettendo un messaggio di posta elettronica.
- **Capacità di esecuzione in remoto:** un worm esegue una copia di se stesso su un altro sistema.
- **Capacità di eseguire operazioni di connessione remota:** un worm si connette a un sistema remoto come un normale utente e successivamente usa i comandi del sistema per copiare se stesso da un sistema a un altro.

La nuova copia del programma worm viene poi eseguita sul sistema remoto dove prosegue la propria diffusione in maniera analoga, in aggiunta alle funzioni distruttive svolte in quel sistema.

Un worm di rete presenta le medesime caratteristiche di un virus informatico: una fase latente, una fase di propagazione, una fase di innesco e una fase di esecuzione. La fase di propagazione in genere svolge le seguenti funzioni.

1. Ricerca di ulteriori sistemi da infettare esaminando le tabelle dell'host o archivi simili contenenti indirizzi di sistemi remoti.
2. Instaurazione di una connessione con un sistema remoto.
3. Copia di se stesso sul sistema remoto, facendo in modo che la copia venga eseguita.

Un worm di rete può anche tentare di stabilire se un sistema è già stato infettato prima di procedere ad auto-copiarsi sul sistema stesso. Su un sistema multi-programmato può anche mascherare la propria presenza auto-denominandosi come un processo di sistema o impiegando un nome che passi inosservato a un operatore di sistema.

Come con i virus, i programmi worm di rete sono difficili da contrastare.

Worm di Morris

Prima dell'attuale generazione di worm, il più conosciuto era quello rilasciato su Internet da Robert Morris nel 1998. Progettato per diffondersi sui sistemi UNIX, usava svariate tecniche di propagazione. Quando una copia iniziava l'esecuzione, la sua prima attività era quella di scoprire le altre macchine conosciute dall'host sul quale si trovava e che avrebbero accettato informazioni provenienti da questo. Questo worm procedeva esaminando molti elenchi e tabelle, comprese le tabelle di sistema che dichiaravano quali altri macchine erano considerate fidate da questo host, i file per la spedizione della posta elettronica, le tabelle attraverso le quali gli utenti attribuivano il permesso di accedere ad account remoti, e un programma che riportava lo stato delle connessioni di rete. Per ciascuno degli host scoperti, il worm provava molte tecniche per insinuarsi.

1. Tentava di accedere e registrarsi a un host remoto come un utente legittimo. Con questo metodo, in primo luogo, cercava di "scassinare" il file locale delle password e poi utilizzava le password scoperte e i corrispondenti ID degli utenti. L'assunzione era che molti utenti avrebbero usato la stessa password su sistemi differenti. Per ottenere le password, questo worm eseguiva un programma password-craking che provava:
 - a. ciascun nome dell'account dell'utente e semplici permutazioni di questo.
 - b. una lista di 432 password già pronte che Morris pensava fossero probabili candidate.
 - c. tutte le parole nel sistema di directory locale.
2. Sfruttava un buco nel protocollo finger che riportava in che posto si trova l'utente remoto.
3. Sfruttava una trapdoor nell'opzione di individuazione e recupero degli errori del processo remoto che riceve e inviava la posta.

Se qualcuno di questi attacchi aveva successo, questo worm otteneva la comunicazione con l'interprete dei comandi del sistema operativo. Inviava poi a questo interprete un breve programma di avvio, emetteva un comando che mandava in esecuzione quel programma e poi si disconnetteva. Il programma di avvio chiamava poi il programma padre e scaricava il resto del programma worm. Il nuovo worm veniva poi eseguito.

Attacchi recenti tramite worm

L'era contemporanea delle minacce worm iniziò con il rilascio del cosiddetto worm Code Red (Codice Rosso) nel luglio del 2001. Code Red sfruttava un buco nella sicurezza di Microsoft Internet Information Server (IIS) per penetrare e diffondersi, e disabilitava anche il controllore dei file di sistema di Windows. Questo worm sondava indirizzi IP casuali per diffondersi sugli altri host. Durante un certo periodo di tempo si diffondeva soltanto, iniziava poi un attacco di negazione del servizio contro un sito web della pubblica amministrazione, inondandolo con pacchetti provenienti da numerosi host. Poi sospendeva le attività e si riattivava periodicamente. Nella seconda ondata di attacchi, Code Red infettò quasi 360.000 server in 14 ore. Oltre alla distruzione provocata sui server designati come bersaglio, Code Red può consumare una quantità enorme di capacità di Internet, interrompendone il servizio.

Code Red II è una variante che ha come bersaglio Microsoft IIS. In aggiunta, questo worm più recente installa una backdoor che permette a un hacker di dirigere le attività dei computer che ne sono vittime.

Alla fine del 2001, apparve un worm più variabile, noto come Nimda, che si diffonde tramite svariati meccanismi

- Da client a client via posta elettronica.
- Da client a client tramite condivisioni aperte sulla rete.
- Da server web a client tramite la navigazione in siti web compromessi.
- Da client a server web tramite le esplorazioni attive, per trovare le diverse vulnerabilità nell'attraversamento delle directory di Microsoft IIS 4.0/5.0, e tramite il loro sfruttamento.
- Da client a server web attraverso la scansione per trovare backdoor che worm di tipo Code Red II si sono lasciati dietro.

Code Red II modifica i documenti web (ad esempio: i file .htm, .html e .asp) e certi file eseguibili che ha trovato sul sistema infettato, e crea numerose copie di se stesso assegnandogli diversi nomi di file.

All'inizio del 2003, comparve un nuovo worm detto SQL Slammer, che sfruttava una vulnerabilità di buffer overflow di Microsoft SQL server. Slammer era estremamente comune. Alla fine del 2003 arrivò Sobig.f, che sfruttava i server proxy aperti per trasformare le macchine infette in motori di spam. Nel momento della sua massima diffusione, Sobig.f – stando a quanto che si dice – contagiò un messaggio ogni 17 e produsse più di un milione di copie di se stesso nelle prime 24 ore.

Mydoom è un worm della posta elettronica di massa che apparve nel 2004. Seguiva la tendenza in crescita di installare una backdoor sui computer infettati, abilitando così gli hacker ad ottenere l'accesso da remoto a dati come password e numeri di carte di credito. Mydoom si replicò fino a 1.000 volte al minuto e, stando a quanto che si dice, inondò Internet con 100 milioni di messaggi infetti in 36 ore.

Stato della tecnologia degli worm

Lo stato dell'arte nella tecnologia degli worm comprende le seguenti tipologie.

- **Multi-piattaforma:** i nuovi worm non sono limitati alle macchine Windows, ma possono attaccare diverse piattaforme, specialmente le varietà di UNIX più comuni.
- **Multi-exploit:** i nuovi worm penetrano nei sistemi in molti modi, usando azioni contro server web, browser, posta elettronica, condivisione di file e altre applicazioni basate sulla rete.
- **Diffusione ultra veloce:** una tecnica per accelerare la diffusione di un worm è condurre un'iniziale esplorazione della rete per accumulare indirizzi Internet di macchine vulnerabili.
- **Polimorfici:** per evitare l'individuazione, superare i vecchi filtri e sgominare l'analisi in tempo reale, gli worm adottano la tecnica del virus polimorfico. Ciascuna copia del

programma worm ha un nuovo codice, generato al volo, usando istruzioni funzionalmente equivalenti e tecniche di cifratura.

- **Metamorfici:** oltre a cambiare il loro aspetto, questi hanno un repertorio di configurazioni di comportamento che vengono scatenate in differenti stadi di propagazione.
- **Veicoli di trasporto:** dato che gli worm possono compromettere rapidamente un gran numero di sistemi, sono ideali per diffondere altri strumenti per attacchi distribuiti, come gli zombie per la negazione del servizio distribuita.
- **Azione del giorno zero:** per ottenere la massima sorpresa e distribuzione, un worm potrebbe sfruttare una vulnerabilità non nota che viene scoperta dalla comunità collettiva di rete quando il programma worm viene lanciato.

10.2 Rimedi contro i virus

Approcci antivirus

La prevenzione rappresenta la soluzione ideale contro i virus: risulta evidente che in primo luogo occorre impedire ai virus di entrare nel sistema. Comunque, anche se la prevenzione può ridurre il numero di attacchi virali andati a buon fine, l'obiettivo di una protezione totale è irraggiungibile. L'approccio migliore da attuare qualora la prevenzione non abbia avuto successo, è il seguente.

- **Rilevazione:** una volta avvenuta l'infezione, rilevarla e localizzare il virus.
- **Identificazione:** compiuta la rilevazione, identificare lo specifico virus che ha infettato il programma.
- **Rimozione:** identificato il virus, rimuoverne tutte le tracce dal programma infetto e ripristinarne lo stato originario. Eliminare il virus da tutti i sistemi infetti in modo che l'infezione non si espanda ulteriormente.

Se la rilevazione va a buon fine, ma non è possibile eseguire l'identificazione o la rimozione, allora l'alternativa è quella di scartare il programma infetto e caricarne nuovamente una versione di backup "pulita".

Gli avanzamenti nella tecnologia di virus e antivirus evolvono rapidamente, di pari passo. I primi virus erano frammenti di codice relativamente semplici e potevano essere identificati e ripuliti con semplici pacchetti software antivirus. Con l'evoluzione della tecnologia, i virus e – di conseguenza – anche i software antivirus sono diventati più complessi e sofisticati. [STEP93] identifica quattro generazioni di software antivirus.

- Prima generazione: semplici programmi di scansione.
- Seconda generazione: programmi di scansione di tipo euristico.
- Terza generazione: trap di attività.
- Quarta generazione: protezione completa.

Un programma di scansione di **prima generazione** necessita di una signature per effettuare l'identificazione di un virus. Il virus può contenere wildcards, ma ha sostanzialmente la stessa struttura e le stesse configurazioni di bit in tutte le sue copie. Questi programmi di scansione si limitano a rilevare virus conosciuti sulla base delle loro signature. Un altro tipo di programmi di scansione di prima generazione tiene traccia della lunghezza dei programmi e controlla le variazioni di lunghezza.

Un programma di scansione di **seconda generazione** non si basa su una specifica signature. Piuttosto, usa regole euristiche per ricercare possibili infezioni virali. Una categoria di programmi di scansione di questo tipo ricerca frammenti di codice che spesso sono associati ai virus. Ad esempio, il programma di scansione può ricercare l'inizio di un ciclo di cifratura utilizzato nei virus polimorfi e scoprire la chiave di cifratura. Una volta scoperta tale chiave, il programma di scansione può decifrare il virus per l'identificazione e poi rimuovere l'infezione e ripristinare il programma originario.

Un'altra tipologia di programmi di scansione di seconda generazione si fonda sulla verifica di integrità. Si appone un valore di checksum a ciascun programma. Se un virus lo infetta senza cambiarne il checksum, allora la verifica di integrità rileverà il cambiamento. Per contrastare un virus sofisticato al punto da modificare anche il valore di checksum in fase di infezione di un programma, si può utilizzare una funzione hash cifrata. La chiave di cifratura è memorizzata separatamente dal programma in modo che il virus non possa generare un nuovo codice hash e cifrarlo. Utilizzando una funzione hash al posto di una più semplice checksum, si impedisce al virus di adattare il programma a produrre lo stesso codice hash posseduto prima dell'infezione.

I programmi di **terza generazione** sono programmi residenti in memoria che identificano un virus in base alle sue azioni piuttosto che rispetto alla sua struttura all'interno di un programma infetto. Tali programmi hanno il vantaggio di non richiedere la definizione di signature ed euristiche per una vasta gamma di virus. In questo caso, è necessario solamente identificare l'insieme – ridotto – di azioni sintomatiche di possibile infezione in corso e quindi intervenire.

I prodotti di **quarta generazione** sono pacchetti software contenenti una vasta gamma di tecniche antivirus usate congiuntamente, fra cui componenti per la scansione e trap di attività. Inoltre, un prodotto di quarta generazione comprende funzionalità di controllo dell'accesso, per limitare le capacità del virus di penetrare nel sistema e di aggiornare file per passare l'infezione.

Con i pacchetti di quarta generazione, si ha una strategia di difesa più estesa, con misure di sicurezza di carattere più generale per i sistemi di elaborazione.

Tecniche antivirus di tipo avanzato

Gli approcci e i prodotti antivirus sono sempre più numerosi e sofisticati: i due più importanti sono qui di seguito descritti.

Generic decryption

La tecnologia Generic decryption (GD) permette al programma antivirus di rilevare facilmente anche i virus polimorfi più complessi, mantenendo nel contempo rapidità di scansione [NACH97]. Quando viene eseguito un file contenente un virus polimorfo, il virus

deve decifrare se stesso per attivarsi. Per scoprire tale struttura, i file eseguibili sono mandati in esecuzione attraverso il programma di scansione GD, contenente i seguenti elementi.

- **Emulatore di CPU:** elaboratore virtuale basato su software. L'emulatore interpreta le istruzioni di un file eseguibile invece di eseguirle sul processore sottostante. L'emulatore comprende versioni software di tutti i registri e di altri componenti hardware del processore, in modo che quello reale non venga toccato dai programmi interpretati dall'emulatore.
- **Programma di scansione della signature del virus:** modulo che effettua la scansione del codice sotto analisi per ricercare signature di virus note.
- **Modulo di controllo dell'emulazione:** controlla l'esecuzione del codice sotto analisi.

All'inizio di ogni simulazione, l'emulatore inizia a interpretare le istruzioni nel codice sotto analisi, una alla volta. Quindi, se il codice contiene una procedura che decifra e poi mostra il virus, quel codice viene interpretato. In effetti, il virus "collabora" con il programma antivirus palesando la propria presenza. Periodicamente, il modulo di controllo interrompe il processo di interpretazione per effettuare la scansione del codice sotto analisi ricercando signature di virus.

Durante l'interpretazione, il codice sotto analisi non causa danni all'ambiente del PC, in quanto l'interpretazione avviene in ambiente completamente controllato.

Il problema di progettazione più complesso relativo a un programma di scansione GD è connesso alla determinazione della durata di ciascuna interpretazione. In genere, gli elementi del virus sono attivati quasi subito dopo l'inizio dell'esecuzione del programma. Ma potrebbe non essere sempre così. Quanto più a lungo il programma di scansione emula un particolare programma, tanto più elevata è la probabilità di smascherare i virus nascosti. Tuttavia, il programma antivirus può occupare solo una limitata quantità di tempo e di risorse per evitare lamentele da parte dell'utenza.

Digital immune system

Si tratta di un approccio completo per la protezione da virus sviluppato da IBM [KEPH97a, KEPH97b]. La motivazione che ha portato allo sviluppo di questo sistema è stata la crescente minaccia rappresentata dalla propagazione di virus tramite Internet.

Originariamente la minaccia dei virus era caratterizzata da una diffusione relativamente lenta sia di nuovi virus sia di nuove versioni di virus esistenti (mutazioni). Il software antivirus veniva in genere aggiornato su base mensile, sufficiente per tenere sotto controllo il problema. Inoltre, Internet giocava un ruolo relativamente ridotto nella diffusione di virus. È stato però evidenziato [CHES97] come due principali tendenze nella tecnologia Internet hanno avuto un impatto crescente nella propagazione dei virus in questi ultimi anni.

- **Sistemi integrati di posta:** sistemi come Lotus Notes e Microsoft Outlook rendono molto semplice inviare qualsiasi cosa a qualsiasi persona e lavorare con gli oggetti ricevuti mediante la posta.
- **Sistemi basati su software mobile:** funzionalità quali Java e ActiveX consentono lo spostamento di programmi da un sistema all'altro.

In risposta alle minacce poste da queste funzionalità basate su Internet, IBM ha sviluppato un prototipo di digital immune system. Questo sistema amplia l'uso dell'emulazione di programmi (descritto nel paragrafo precedente), e fornisce un'emulazione di uso generale e un sistema di rilevazione di virus. Obiettivo di questo sistema è fornire tempi di risposta rapidi in modo che i virus possano essere eliminati non appena introdotti nel sistema. Quando un nuovo virus fa il suo ingresso in un'organizzazione, il sistema lo cattura automaticamente, lo analizza, aggiunge identificazione e protezione, lo rimuove e passa le informazioni sul virus in questione ai sistemi che eseguono AntiVirus IBM, in modo da poterlo scoprire prima che possa essere eseguito altrove. La Figura 10.4 illustra le tipiche fasi del funzionamento di digital immune system.

1. Un programma di monitoraggio su ogni PC usa una gamma di euristiche basate sul comportamento del sistema, su cambiamenti sospetti a programmi o su signature per inferire la presenza di un virus. Il programma di controllo inoltra una copia di qualche programma ritenuto infetto a una macchina con ruolo di amministratore nell'organizzazione.
2. Tale macchina effettua la cifratura del campione e lo invia a una macchina centrale di analisi dei virus.
3. Quest'ultima macchina crea un ambiente in cui poter eseguire in maniera sicura il programma infetto. A tale scopo, si utilizzano tecniche di emulazione, oppure si crea un ambiente protetto in cui il programma sospetto possa essere eseguito e monitorato. La macchina di analisi dei virus produce poi una "ricetta" per l'identificazione e la rimozione del virus rilevato.
4. La ricetta viene rimandata alla macchina di amministrazione.

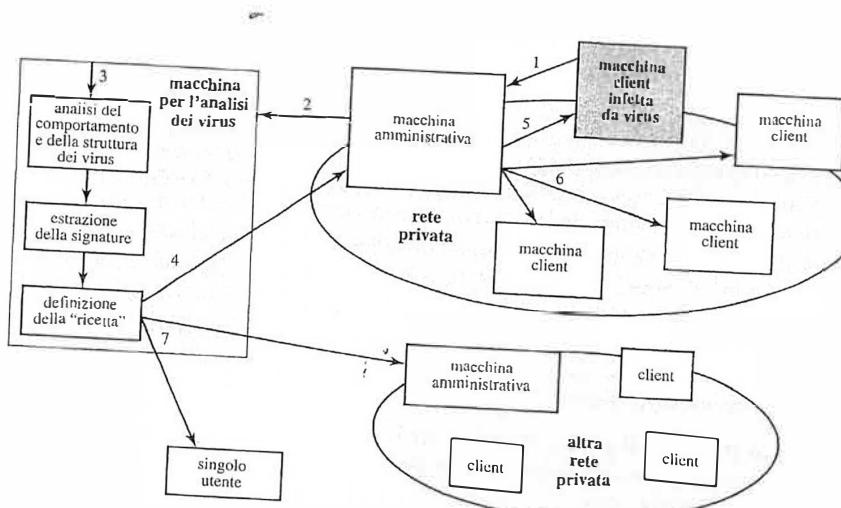


Figura 10.4 Digital immune system.

5. Questa inoltra la ricetta al client infetto.
6. La ricetta viene inoltrata anche ad altri client dell'organizzazione.
7. Gli abbonati al sistema, sparsi nel mondo, ricevono aggiornamenti regolari del software antivirus per la protezione contro il nuovo virus.

Il successo di digital immune system dipende dalla capacità della macchina di analisi di scoprire nuove variazioni di virus e virus innovativi. Analizzando e controllando costantemente i virus trovati, dovrebbe essere possibile mantenere continuamente aggiornato il software del digital immune system rispetto alle possibili minacce.

Software a blocco del comportamento (behavior-blocking)

A differenza dei programmi di scansione, che si basano su tecniche euristiche o sulle signature dei software dolosi, un software a blocco del comportamento si integra con il sistema operativo sul computer host e monitora il comportamento del programma in tempo reale per individuare azioni dannose. Blocca, quindi, le potenziali azioni nocive prima che abbiano la possibilità di colpire il sistema. Le tipologie dei comportamenti controllati possono essere le seguenti:

- tentativi di aprire, vedere, cancellare e/o modificare file;
- tentativi di formattare i dischi o altre azioni irrimediabili sui dischi;
- modifiche alla logica dei file eseguibili e delle macro;
- modifiche a configurazioni critiche del sistema, come la configurazione di avvio;
- scripting dei client di posta elettronica e della messaggistica istantanea per inviare contenuti eseguibili;
- avvio di connessioni di rete.

Se i sistemi a blocco del comportamento rilevano che un programma, appena è in esecuzione, sta intraprendendo comportamenti dannosi, allora possono bloccare tali comportamenti in tempo reale e/o terminare il software offensivo. Questo offre un vantaggio fondamentale rispetto a tecniche affermate di individuazione antivirus, come quelle di rilevazione delle signature o alle tecniche euristiche. Mentre esistono, letteralmente, trillions di modi differenti per confondere e risistemare le istruzioni di un virus o di un worm, molti dei quali eviteranno di essere individuati da un analizzatore delle signature o da un sistema euristico, alla fine il codice doloso deve fare una ben definita richiesta al sistema operativo. Dato che il sistema a blocco del comportamento può intercettare tutte le richieste come queste, può identificare e bloccare le azioni dannose indipendentemente da quanto si presenti nascosta la logica del programma.

L'abilità di osservare il software appena va in esecuzione e in tempo reale chiaramente offre un enorme vantaggio al sistema a blocco del comportamento, tuttavia presenta anche un lato negativo. Dato che il codice dannoso deve realmente girare sulla macchina bersaglio prima che tutti i suoi comportamenti possano essere identificati, può causare una grande quantità di danni al sistema prima che venga individuato e bloccato dal sistema a blocco del comportamento. Per esempio, un nuovo virus potrebbe mettere alla rinfusa molti file ap-

parentemente poco importanti sul disco rigido prima di infettare un singolo file e venire bloccato. Anche se l'infezione effettiva è stata bloccata, l'utente potrebbe non essere in grado di localizzare i propri file, subendo una perdita di produttività o forse peggio.

10.3 Attacchi di negazione del servizio distribuiti

Gli attacchi di negazione del servizio distribuiti (DDoS, *distributed denial of service*) rappresentano una significativa minaccia di sicurezza per le società e sembra che questa minaccia sia in crescita [VIJA02]. In uno studio del 2001 – relativo al monitoraggio, per un periodo di tre settimane, di società di commercio elettronico, come Amazon o Hotmail, fino a piccoli ISP e collegamenti dial-up – gli investigatori osservarono più di 12.000 attacchi contro più di 5.000 bersagli [MOOR01]. Gli attacchi DDoS rendono i sistemi di elaborazione inaccessibili inondando i server, le reti e anche i sistemi utente con traffico inutile, in modo che gli utenti legittimi non possano più aver accesso a quelle risorse. In un tipico attacco DDoS vengono radunati un gran numero di host compromessi per inviare pacchetti inutili. Negli ultimi anni, i metodi e gli strumenti di attacco sono diventati sofisticati, efficienti ed è più difficile rintracciare i reali attaccanti, mentre le tecnologie di difesa non sono state in grado di opporsi ad attacchi su larga scala [CHAN02].

Un attacco di negazione del servizio è un tentativo di impedire ai legittimi utenti l'uso di un servizio. Quando questo attacco proviene da un host singolo o da un nodo di rete, allora lo si definisce semplicemente un attacco DoS. Gli attacchi DDoS pongono una minaccia più seria. In un attacco DDoS un attaccante è in grado di reclutare molti host in Internet per lanciare simultaneamente, o in maniera coordinata, un attacco verso il bersaglio. Questo paragrafo riguarda gli attacchi DDoS. In primo luogo, si osserverà la natura e il tipo di attacchi, successivamente, si esamineranno i mezzi con i quali un attaccante può reclutare una rete di host per lanciare l'attacco. Infine si considereranno i rimedi.

Descrizione degli attacchi DDoS

Un attacco DDoS è un tentativo di consumare le risorse del bersaglio in modo che non sia in grado di fornire il servizio. Un modo di classificare gli attacchi DDoS è in termini del tipo di risorsa che viene consumata. Parlando in generale, la risorsa consumata è o una risorsa interna all'host sul sistema bersaglio o la capacità di trasmissione dei dati in una rete locale alla quale il bersaglio è collegato.

Un semplice esempio di un **attacco alle risorse interne** è quello di SYN flood. Nella Figura 10.5a sono mostrati i passi richiesti.

1. L'attaccante prende il controllo di più host su Internet, istruendoli a contattare il server web bersaglio.
2. Gli host slave iniziano a inviare al bersaglio pacchetti TCP/IP di tipo SYN (sincronizzazione/inizializzazione) con informazioni dell'indirizzo IP di ritorno errate.
3. Ciascun pacchetto SYN è una richiesta di aprire una connessione TCP. Per ciascun pacchetto di questo tipo il server web risponde con un pacchetto di SYN/ACK (sincronizzazione/riscontro), provando a stabilire una connessione TCP con l'entità TCP all'indirizzo IP di ritorno errato.

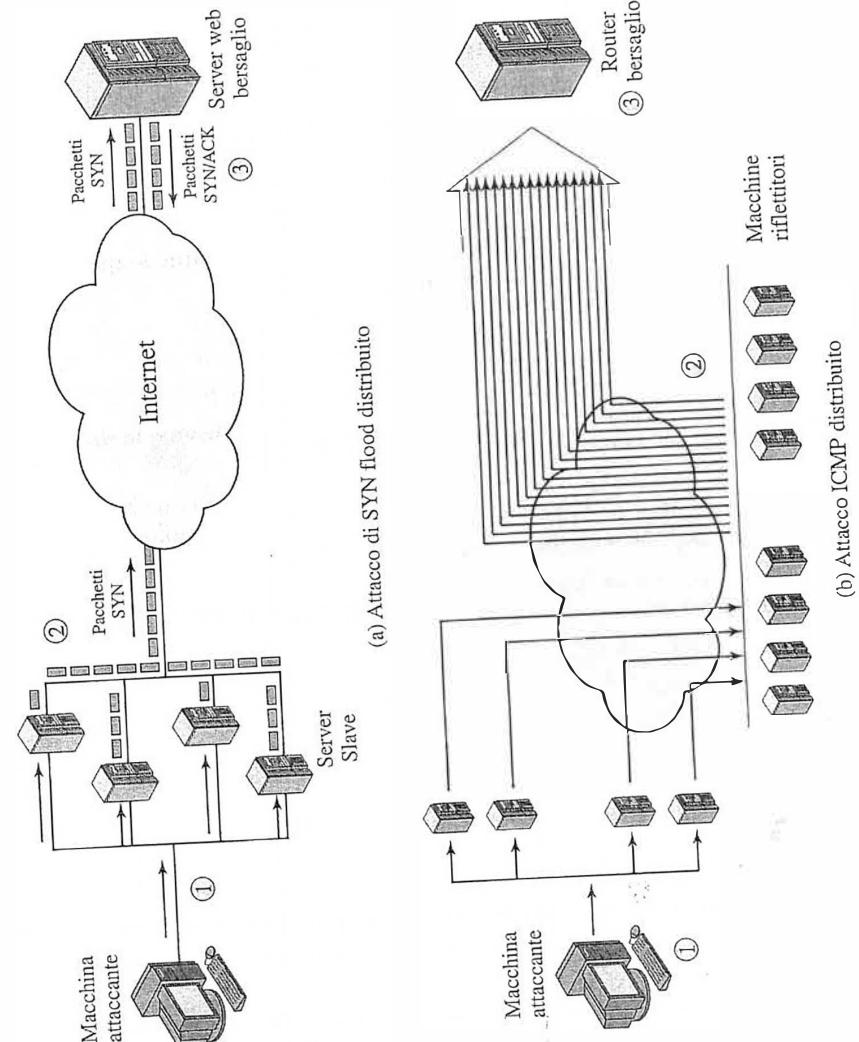


Figura 10.5 Esempi di semplici attacchi DDoS.

rizzo IP falso. Il server web mantiene le strutture dati per ciascuna richiesta SYN, aspettando una risposta e rallenta sempre di più l'esecuzione man mano che aumenta il traffico in arrivo. Il risultato è che le connessioni legittime sono negate, mentre la macchina che ne è vittima sta aspettando di completare le finte connessioni "mezze-aperte".

La struttura dati dello stato TCP è una risorsa interna comune bersaglio di attacchi, ma questo non significa che sia la sola. [CERT01] fornisce i seguenti esempi.

1. In molti sistemi è disponibile un numero limitato di strutture dati per memorizzare informazioni sui processi (identificativi di processo, elementi della tabella dei processi, slot di processi, etc.). Chi vuole effettuare un'intrusione potrebbe essere in grado di esaurire queste strutture dati scrivendo un semplice programma o script che non fa nulla se non creare ripetutamente copie di se stesso.
2. Chi vuole effettuare un'intrusione potrebbe anche tentare di esaurire lo spazio su disco in altri modi, compresi:
 - generare un numero enorme di messaggi di posta;
 - generare intenzionalmente errori che devono essere registrati;
 - mettere file in aree FTP anonime o in aree condivise sulla rete;

Nella Figura 10.5b viene illustrato un esempio di un **attacco che consuma le risorse per la trasmissione dei dati**. Sono richiesti i seguenti passi.

1. Un attaccante prende il controllo di più host su Internet, istruendoli a inviare pacchetti di ECHO ICMP¹, con l'indirizzo IP sorgente contraffatto e pari a quello del bersaglio, a un gruppo di host, i quali agiranno come riflettori, come descritto successivamente.
2. I nodi riflettori ricevono le richieste contraffatte e rispondono mandando pacchetti di risposta all'ECHO al sito bersaglio.
3. Il router del bersaglio è invaso da pacchetti provenienti dal sito dove vi sono i riflettori, in modo tale da non lasciare capacità di trasmissione dati al traffico legittimo.

Un altro modo di classificare gli attacchi DDoS è come attacchi DDoS diretti o tramite riflettori. In un attacco **DDoS diretto** (Figura 10.6a), l'attaccante è in grado di installare software zombie su parecchi siti distribuiti su Internet. Sovente gli attacchi DDoS distribuiti coinvolgono due livelli di macchine zombie: gli zombie master e gli zombie slave. Gli host di entrambe le macchine vengono infettate con codice doloso. L'attaccante coordina e fa partire gli zombie master, che a loro volta coordinano e fanno partire gli zombie slave. L'uso di due livelli di zombie rende più difficile rintracciare l'attacco risalendo fino alla sua sorgente, e fornisce una rete di attaccanti più flessibile.

Un attacco **DDoS tramite riflettori** aggiunge un altro livello di macchine (Figure 10.6b). In questo tipo di attacco, gli zombie slave costruiscono pacchetti richiedendo una risposta che contiene l'indirizzo IP del bersaglio come indirizzo IP della sorgente nell'inten-

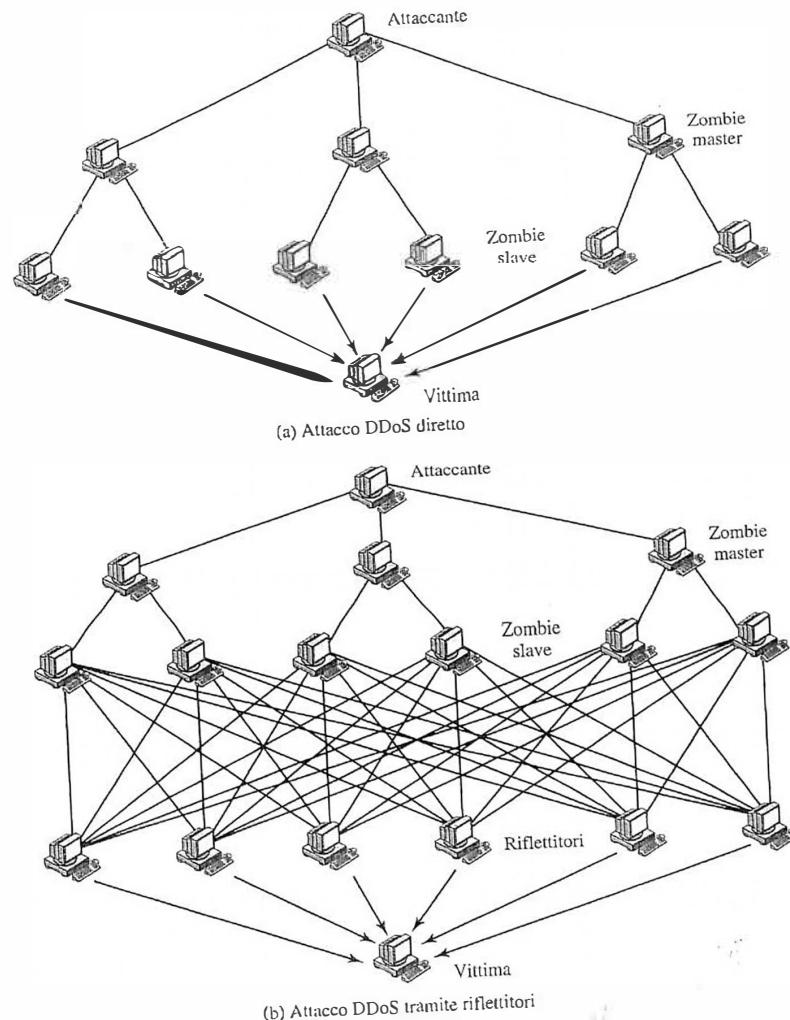


Figura 10.6 Tipi di attacco DDoS basati sull'inondazione con dati delle macchine bersaglio.

stazione del pacchetto IP. Questi pacchetti vengono inviati a macchine non infette: i riflettori. Le macchine non infette rispondono con pacchetti diretti alle macchine bersaglio. Un attacco DDoS tramite riflettori può facilmente coinvolgere più macchine e più traffico di un attacco DDoS diretto e, quindi, può essere maggiormente dannoso.

Inoltre, risulta difficile risalire all'origine dell'attacco o filtrare i pacchetti in quanto l'attacco proviene da macchine non infette disseminate su Internet.

¹ Internet Control Message Protocol (ICMP) è un protocollo a livello IP per lo scambio di pacchetti di controllo tra un router e un host o tra host. Il pacchetto ECHO richiede al ricevente di rispondere con una risposta all'ECHO per verificare che tra le due entità è possibile la comunicazione.

Costruzione di una rete di attacco

Durante la prima fase di un attacco DDoS, l'aggressore infetterà molte macchine tramite software zombie finalizzato a portare avanti l'attacco. Gli elementi essenziali di questa fase sono i seguenti.

1. Il software che può condurre l'attacco DDoS. Il software deve essere in grado di girare su un elevato numero di macchine, di nascondere la sua esistenza, di comunicare con l'attaccante o avere qualche sorta di meccanismo a innesco temporale, e deve essere in grado di lanciare l'attacco verso il bersaglio.
2. Vulnerabilità di un grande numero di sistemi. L'attaccante deve individuare una vulnerabilità che molti amministratori di sistema e utenti individuali non sono riusciti a correggere e che gli consente di installare il software zombie.
3. Una strategia per localizzare le macchine vulnerabili; processo noto come scansione (*scanning*).

Nel processo di scansione, in primo luogo l'attaccante individua svariate macchine vulnerabili e le infetta. In seguito, tipicamente, i software zombie installati sulle macchine infette ripetono lo stesso processo di scansione, fino a che viene creata un'ampia rete distribuita di macchine infette. [MIRK04] elenca i seguenti tipi di strategie di scansione.

- **Casuale:** ciascun host compromesso tenta degli indirizzi casuali nello spazio di indirizzi IP, usando un seme differente. Questa tecnica produce un grande volume di traffico in Internet, che può causare l'interruzione generalizzata delle comunicazioni, anche prima che l'attacco effettivo venga lanciato.
- **Hit-list:** l'attaccante compila in primo luogo una lunga lista di macchine potenzialmente vulnerabili. Questo può essere un processo lento, eseguito in un lungo periodo, per evitare che ci si accorga che è in corso un attacco. Una volta compilata, l'attaccante inizia ad infettare le macchine presenti nella lista. A ciascuna macchina infetta viene fornita una parte della lista da analizzare. Questa strategia dà luogo a un periodo di scansione molto breve che potrebbe rendere difficile l'individuazione che l'infezione sta avvenendo.
- **Topologica:** questo metodo usa le informazioni contenute su una macchina vittima per trovare più host da analizzare.
- **Sottorete Locale:** se un host dietro a un firewall può essere infetto, quell'host allora cercherà bersagli nella propria rete locale. L'host usa la struttura dell'indirizzo di sottorete per trovare altri host che altrimenti sarebbero protetti dal firewall.

Rimedi per gli attacchi DDoS

In generale, esistono tre linee di difesa contro gli attacchi DDoS [CHAN02].

- **Prevenzione e cognizione** (prima dell'attacco): questi meccanismi abilitano la vittima a resistere ai tentativi di attacco senza negare il servizio agli utenti legittimi. Le tecniche comprendono il rispetto delle politiche per il consumo delle risorse e la formatura di risorse di riserva disponibili su richiesta. Inoltre, i meccanismi di prevenzione modificano i sistemi e i protocolli su Internet, per ridurre la possibilità di attacchi DDoS.
- **Rilevazione degli attacchi e filtraggio** (durante l'attacco): questi meccanismi tentano di individuare l'attacco appena inizia e di reagire immediatamente. Questo riduce l'impatto dell'attacco sul bersaglio. La rilevazione implica la ricerca di configurazioni di comportamento sospette, mentre la reazione implica il filtraggio dei pacchetti che probabilmente fanno parte dell'attacco.
- **Risalire alla sorgente dell'attacco e identificazione** (durante e dopo l'attacco): questo è un tentativo di identificare la sorgente dell'attacco come primo passo nella prevenzione degli attacchi futuri. Tuttavia, questo metodo non offre generalmente risultati sufficientemente rapidi, se non addirittura nulli, per attenuare l'attacco in corso.

La sfida per l'imitare gli attacchi DDoS consiste proprio nel numero di modi in cui possono funzionare: quindi, i rimedi contro gli attacchi DDoS devono progredire di pari passo con le minacce.

10.4 Letture consigliate e siti web

Per una completa comprensione dei virus, il libro da leggere è [SZOR05]. Un'altra eccellente trattazione è [HARL01]. Interessanti articoli di rassegna su virus e worm sono [CASS01], [FORR97], [KEPH97] e [NACH97]. [MEIN01] fornisce una buona trattazione di Code Red.

[PATR04] è un meritevole studio sugli attacchi DDoS e [MIRK04] è una completa descrizione delle varietà di attacchi DDoS e dei loro rimedi. [CHAN02] è un'eccellente indagine sulle strategie difensive da attacchi DDoS.

Siti web consigliati

- **AntiVirus Online:** sito dell'IBM con informazioni sui virus.
- **Vmyths:** dedicato allo smascheramento dei virus beffa e per chiarire i malintesi sui virus reali.
- **Attacchi/strumenti DDoS:** lista esaurente di collegamenti e documenti.

10.5 Domande di revisione ed esercizi

Domande di revisione

- 10.1 Qual è il ruolo della compressione nelle operazioni di un virus?
- 10.2 Qual è il ruolo della cifratura nelle operazioni di un virus?
- 10.3 Quali sono le fasi tipiche del funzionamento di un virus o di un worm?
- 10.4 In termini generali, come si propaga un worm?
- 10.5 Che cos'è un digital immune system?
- 10.16 Come funziona un software a blocco del comportamento?
- 10.7 Che cos'è un DDoS?

Esercizi

- 10.1 Il virus nella Figura 10.1 contiene un difetto: quale?
- 10.2 Nasce la questione su come sia possibile sviluppare un programma che può analizzare un pezzo di software per determinare se è un virus. Si consideri un programma D, che si suppone sia in grado di farlo; cioè, per qualsiasi programma P, se viene eseguito D(P), il risultato dà VERO (P è un virus) o FALSO (P non è un virus). Si consideri ora il seguente programma:

```

Programma CV:=
{
    .
    .
}

Programma-principale:=
{
    if D(CV) then goto next;
    else infetta-eseguibile;
}

next:
}

```

Nel precedente programma, "infetta-eseguibile" è un modulo che analizza la memoria dei programmi eseguibili e si replica in quei programmi. Si determini se D può correttamente decidere se CV sia un virus.

Capitolo 11

Firewall

I firewall costituiscono un mezzo efficace per proteggere un sistema locale o una rete di sistemi da attacchi perpetrati tramite la rete, permettendo nel contempo accesso al mondo esterno tramite reti geografiche e Internet.

Il capitolo fornisce, innanzitutto, una panoramica delle funzionalità e dei principi di progettazione alla base della tecnologia dei firewall; successivamente, verrà trattato il problema della sicurezza dei firewall, con particolare riferimento al concetto di sistema trusted (fidato) e a quello di sistema operativo sicuro.

11.1 Principi di progettazione di firewall

Per molte organizzazioni, la connettività tramite Internet non è più solo un'opzione ma è diventata un requisito essenziale in quanto le informazioni e i servizi accessibili tramite Internet costituiscono una risorsa molto importante. Inoltre, i singoli utenti desiderano e necessitano di un accesso a Internet, e quando non è disponibile tramite LAN, utilizzano una linea telefonica per collegarsi con il PC a un fornitore di servizi Internet (ISP, *Internet service provider*). Comunque, se è vero che l'accesso a Internet comporta molti vantaggi, è anche vero che consente al mondo esterno di raggiungere e interagire con le risorse della rete locale. Questo rappresenta una minaccia per qualsiasi organizzazione. Una soluzione potrebbe consistere nel dotare ciascuna workstation e ciascun server sulla rete locale di robusti meccanismi di sicurezza come, ad esempio, meccanismi di protezione dalle intrusioni. Tale approccio non è però conveniente. Si consideri infatti una rete composta da centinaia o anche da migliaia di sistemi, dotati di sistemi operativi eterogenei (ad esempio, varie versioni di UNIX, Windows 95, 98 e NT). Quando si scopre un punto debole rispetto alla sicurezza, ogni sistema che può essere potenzialmente influenzato deve essere aggiornato per ovviare a tale inconveniente. La soluzione alternativa, oggi sempre più accettata, è costituita dall'utilizzo di firewall. Un firewall è inserito tra la rete locale e Internet al fine di stabilire un collegamento controllato e innalzare una barriera di sicurezza tra i due. Lo scopo di tale barriera è proteggere la rete locale da attacchi provenienti da Internet e fornire un solo punto di ingresso dove si possono collocare gli opportuni meccanismi di sicurezza e audit. Un firewall può essere formato da un singolo calcolatore o da più sistemi che cooperano per effettuare le funzioni cui il firewall è preposto.

In questo paragrafo saranno, in primo luogo, illustrate le caratteristiche generali dei firewall, poi analizzeremo i tipi oggi più utilizzati di firewall, per esaminare, infine, le loro più comuni configurazioni.

Caratteristiche dei firewall

[BELL94b] elenca i seguenti obiettivi di progetto per un firewall.

1. Tutto il traffico dall'interno verso l'esterno, e viceversa, deve passare attraverso il firewall. Questo requisito si ottiene bloccando fisicamente tutti gli accessi alla rete locale, ad eccezione dell'accesso tramite firewall. Per realizzare ciò sono possibili diverse configurazioni che verranno illustrate più avanti in questo paragrafo.
2. L'unico traffico cui è permesso il passaggio attraverso il firewall è quello autorizzato dalla politica di sicurezza locale. Possono essere utilizzati diversi tipi di firewall che realizzano diverse politiche di sicurezza, come si spiegherà più avanti in questo paragrafo.
3. Il firewall stesso è immune alle penetrazioni. Questo requisito implica l'utilizzo di un sistema trusted dotato di un sistema operativo sicuro. Questo argomento verrà trattato nel Paragrafo 11.2.

[SMIT97] elenca quattro tecniche di carattere generale utilizzate dai firewall per effettuare il controllo dell'accesso e realizzare le politiche di sicurezza locali. Inizialmente, i firewall erano soprattutto dedicati al controllo dei servizi ma oggi forniscono protezione per tutti i quattro aspetti di seguito elencati.

- **Controllo dei servizi.** Stabilisce a quali tipi di servizi Internet si può accedere, sia all'interno sia all'esterno. Il firewall può attuare un filtraggio del traffico sulla base dell'indirizzo IP e del numero di porta TCP; può, inoltre, fornire software che agisce da proxy, che riceve e interpreta ogni richiesta di servizio prima di inoltrarla; oppure può ospitare il software del server stesso come, ad esempio, i servizi di posta elettronica e i servizi web.
- **Controllo della direzione.** Stabilisce la direzione secondo cui particolari richieste di servizio possono essere avviate e inoltrate attraverso il firewall.
- **Controllo utente.** Regola l'accesso a un servizio sulla base dell'utente che ha effettuato la richiesta. Questo tipo di funzionalità è di solito applicata a utenti all'interno dell'area protetta dal firewall (utenti locali). Tale tipo di controllo può anche essere applicato a traffico proveniente da utenti esterni; quest'ultima tipologia di utilizzo richiede una qualche forma di meccanismo di autenticazione sicura quale, ad esempio, IPSec (Capitolo 6).
- **Controllo del comportamento.** Controlla come sono utilizzati particolari servizi. Ad esempio, il firewall può effettuare un filtraggio della posta elettronica per evitare fenomeni di spamming, o può consentire l'accesso dall'esterno solo a una parte delle informazioni sul server web locale.

Prima di analizzare in dettaglio le tipologie di firewall e le relative configurazioni, è opportuno riassumerne i servizi e le funzionalità.

1. Un firewall definisce un singolo punto di entrata che mantiene gli utenti non autorizzati al di fuori della rete protetta, proibisce a servizi potenzialmente vulnerabili di en-

trare o di uscire dalla rete, e fornisce protezione contro diverse tipologie di attacchi di rete (ad esempio, spoofing degli indirizzi IP). L'utilizzo di un singolo punto di entrata rende più agevole la gestione della sicurezza in quanto le capacità di sicurezza sono consolidate all'interno di un singolo sistema o di un insieme di sistemi.

2. Un firewall costituisce un punto di riferimento per effettuare il monitoraggio di eventi legati alla sicurezza. All'interno del firewall si possono realizzare meccanismi di audit e di notifica di possibili violazioni.
3. Un firewall costituisce una piattaforma adatta ad alcune funzioni Internet non specificatamente connesse alla sicurezza. Tali funzioni includono la traduzione degli indirizzi di rete, per trasformare indirizzi locali in indirizzi Internet, e funzioni di gestione di rete, quali funzioni di audit o di monitoraggio dell'utilizzo di Internet.
4. Un firewall può costituire la piattaforma per IPSec. Utilizzando la modalità tunnel, descritta nel Capitolo 6, il firewall può essere impiegato per la realizzazione di reti private virtuali.

I firewall presentano anche alcune limitazioni.

1. I firewall non possono assicurare protezione contro attacchi che riescono ad aggirarli. I sistemi interni possono avere la possibilità di effettuare chiamate all'esterno per connettersi a un ISP. Inoltre, una LAN interna potrebbe essere dotata di una serie di modem che consentono di connettersi alla rete locale dall'esterno.
2. Il firewall non fornisce protezione da minacce interne, quali quelle rappresentate da un impiegato insoddisfatto o da un impiegato che coopera involontariamente con un avversario esterno.
3. Il firewall non fornisce protezione rispetto al trasferimento di programmi o file contenenti virus. Tale limite è una diretta conseguenza della varietà di sistemi operativi e di applicazioni presenti all'interno dell'area protetta dal firewall, che rende poco pratica – se non addirittura impossibile – la scansione di tutti i file, dei messaggi di posta elettronica e dei vari messaggi in arrivo alla ricerca di virus.

Tipologie di firewall

La Figura 11.1 illustra le tre tipologie comuni di firewall: filtri di pacchetti, gateway a livello applicazione e gateway a livello circuito.

Router a filtraggio di pacchetti

Un router a filtraggio di pacchetti applica un insieme di regole a ogni pacchetto IP in ingresso e in uscita e, sulla base della valutazione di tali regole, scarta oppure inoltra il pacchetto. Solitamente il router è configurato in modo da filtrare i messaggi in entrambe le direzioni (da e verso la rete interna). Le regole di filtraggio sono basate sulle informazioni contenute in un pacchetto di rete.

- **Indirizzo IP di sorgente:** l'indirizzo IP del sistema che ha dato origine al pacchetto IP (ad esempio: 192.178.1.1).

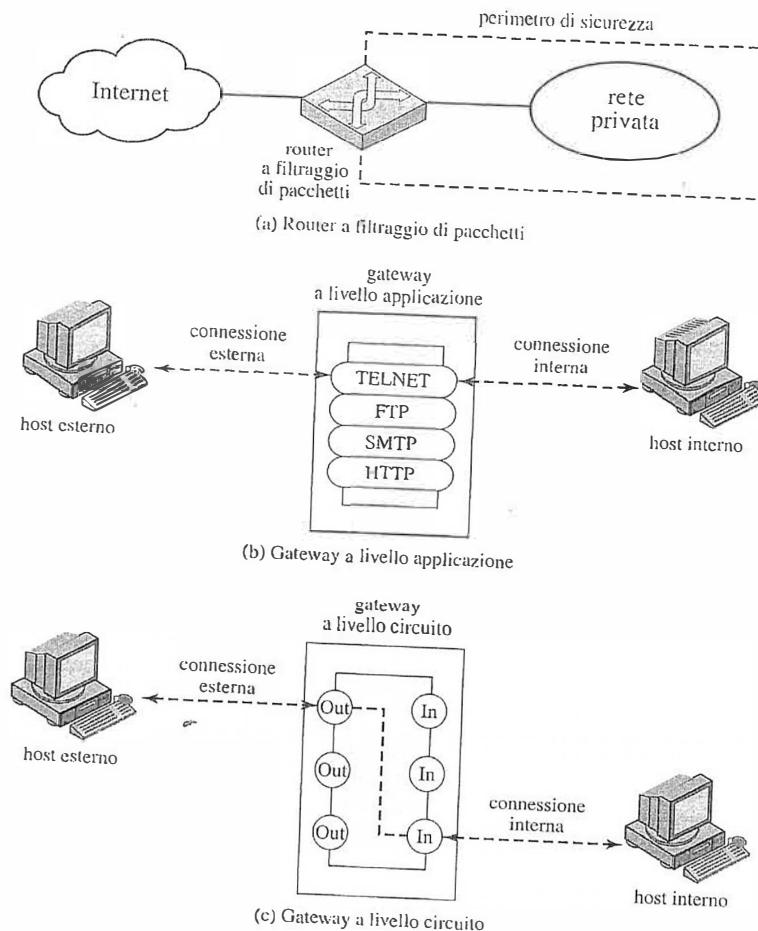


Figura 11.1 Tipologie di firewall.

- **Indirizzo IP di destinazione:** l'indirizzo IP del sistema che il pacchetto IP sta cercando di raggiungere (ad esempio: 192.178.1.2).
- **Indirizzi di livello di trasporto della sorgente e della destinazione:** i numeri di porta a livello di trasporto (ad esempio: TCP o UDP) che definiscono le applicazioni, come SNMP o TELNET.
- **Il campo protocollo IP:** definisce il protocollo di trasporto.
- **Interfaccia:** per un router con tre o più porte, indica da quale interfaccia del router il pacchetto proviene o a quale interfaccia del router il pacchetto è destinato.

Il filtraggio di pacchetti è di solito realizzato come un insieme di regole che si basano su verifiche di alcuni campi nell'intestazione IP o TCP. Se il pacchetto corrisponde a una delle regole, questa viene invocata per stabilire se il pacchetto deve essere inoltrato oppure scartato. Se non esiste corrispondenza con alcuna delle regole, viene eseguita un'azione di default. Questa può essere scelta in base a due politiche.

- **Default = scartare.** Ciò che non è espressamente permesso viene proibito.
- **Default = inoltrare.** Ciò che non è espressamente proibito viene permesso.

La prima politica è la più conservativa. Inizialmente tutto è bloccato, e i servizi devono essere aggiunti caso per caso. Tale politica ha un maggiore impatto presso gli utenti che possono percepire il firewall come un impedimento. La seconda politica aumenta la facilità di utilizzo da parte degli utenti ma fornisce un minor grado di sicurezza; adottando tale politica, l'amministratore della sicurezza deve reagire a ogni nuova minaccia di sicurezza, al momento in cui si manifesta.

La Tabella 11.1, tratta da [BELL94b], fornisce alcuni esempi di insiemi di regole di filtraggio di pacchetti. Per ogni insieme, le regole sono applicate dall'alto verso il basso. Il simbolo "*" in un campo rappresenta una *wildcard* per indicare corrispondenza con qualsiasi cosa. Si assume che la politica adottata sia default = scartare.

- A. È permessa la posta dall'interno (la porta 25 è per SMTP in arrivo), ma solo verso un host gateway. La posta da un particolare host esterno, SPIGOT, è bloccata perché si è osservato che tale host invia spesso file di dimensioni troppo elevate in messaggi di posta elettronica.
- B. Rappresenta una dichiarazione esplicita della politica di default. Tutti gli insiemi di regole includono implicitamente questa come ultima regola.
- C. Questo insieme di regole specifica che ogni host interno può inviare posta all'esterno. Un pacchetto TCP avente come destinazione la porta 25 è instradato al server SMTP sulla macchina di destinazione. Il problema di questa regola è che l'utilizzo della porta 25 per la ricezione SMTP è solo quella di default; una macchina esterna potrebbe essere configurata in modo da avere altre applicazioni collegate alla porta 25. La formulazione di tale regola non evita che un avversario possa ottenere accesso alle macchine interne inviando pacchetti aventi un numero di porta sorgente TCP pari a 25.
- D. Questo insieme di regole supera i limiti dell'insieme di regole al punto C. Tali regole sfruttano una delle caratteristiche delle connessioni TCP. Quando viene stabilita una connessione TCP, il flag ACK posto nel segmento TCP è inizializzato per segnalare l'avvenuta ricezione di segmenti inviati dall'altra parte. Questo insieme di regole consente, quindi, il transito ai pacchetti IP aventi come indirizzo IP sorgente l'indirizzo di un host che fa parte della lista di host interni stabiliti e come destinazione la porta TCP numero 25. Tale insieme di regole consente, inoltre, il transito a pacchetti in arrivo aventi come porta sorgente la porta 25 e che contengono il flag ACK nel segmento TCP. Si noti come sia compito dei sistemi sorgente e destinazione la definizione esplicita di tali regole.

	azione	host interno	porta	host esterno	porta	commento
A	bloccare	*	*	SPIGOT	*	host non affidabile
	azione	host interno	porta	host esterno	porta	commento
B	bloccare	*	*	*	*	connessione alla porta SMTP interna
	azione	host interno	porta	host esterno	porta	commento
C	consentire	*	*	*	*	default
	azione	host interno	porta	host esterno	porta	commento
D	consentire	{host interni}	*	*	*	connessione porta SMTP esterna
	azione	src	porta	dest	porta	flag
E	consentire	*	25	*	*	ACK
	azione	src	porta	dest	porta	flag
	consentire	{host interni}	*	*	*	risposte
	consentire	*	25	*	*	ACK
	consentire	*	*	*	*	traffico non indirizzato a server
	azione	src	porta	dest	porta	flag
	consentire	{host interni}	*	*	*	chiamate verso l'esterno
	consentire	*	*	*	*	risposte
	consentire	*	*	*	*	traffico non indirizzato a server

Tabella 11.1 Esempi di regole di filtraggio di pacchetti.

E. Questo insieme di regole costituisce un approccio alla gestione delle connessioni FTP. Con FTP vengono utilizzate due connessioni TCP: una connessione di controllo, per iniziare il trasferimento dei file, e una connessione dati, per attuare effettivamente il trasferimento. La connessione dati utilizza un numero di porta diverso, dinamicamente assegnato per il trasferimento. La maggior parte dei server – e, quindi, la maggior parte dei bersagli degli avversari – utilizzano porte con numeri bassi; la maggior parte delle chiamate indirizzate verso l'esterno tendono a usare porte con numeri più alti, generalmente superiori a 1023. Questo insieme di regole consente quindi:

- pacchetti creati internamente;
- pacchetti di risposta a una connessione iniziata da una macchina interna;
- pacchetti destinati a una porta con un numero alto su una macchina interna.

Tale schema presuppone che i sistemi siano configurati in modo che si utilizzino soltanto i numeri di porta appropriati.

L'insieme di regole al punto E mette in evidenza la difficoltà di gestire le applicazioni a livello di filtraggio di pacchetti. Un modo alternativo di trattare applicazioni quali FTP e simili è mediante un gateway a livello applicazione, descritto più avanti in questo paragrafo.

Uno dei vantaggi del router a filtraggio di pacchetti è la sua semplicità. Inoltre, il filtraggio dei pacchetti avviene di solito molto velocemente e in modo trasparente all'utente. [WACK02] elenca i seguenti punti deboli dei firewall a filtraggio di pacchetti.

- Dato che i firewall a filtraggio di pacchetti non esaminano i dati dei livelli più alti, non possono evitare attacchi che sfruttano vulnerabilità o funzioni specifiche delle applicazioni. Ad esempio, un firewall a filtraggio di pacchetti non può bloccare i comandi specifici dell'applicazione: se un firewall a filtraggio di pacchetti consente l'accesso a una certa applicazione, tutte le funzioni disponibili all'interno di quella applicazione saranno consentite.
- A causa delle informazioni limitate disponibili al firewall, le funzionalità di registrazione presenti in un firewall a filtraggio di pacchetti sono limitate. Le registrazioni di un filtro di pacchetti contengono le stesse informazioni usate per prendere decisioni sul controllo degli accessi (indirizzo sorgente, indirizzo di destinazione e tipo di traffico).
- La maggior parte dei firewall a filtraggio di pacchetti non supporta gli schemi avanzati di autenticazione utente. Di nuovo, questa limitazione è per lo più dovuta alla mancanza di funzionalità a livello più elevato da parte del firewall.
- Sono generalmente vulnerabili agli attacchi e alle azioni che si avvantaggiano dei problemi all'interno della specifica e della pila protocolare TCP/IP, come la **contraffazione degli indirizzi a livello IP** (*network layer address spoofing*). Molti firewall a filtraggio di pacchetti non possono individuare un pacchetto di rete nel quale le informazioni di indirizzamento a livello 3 OSI sono state alterate. Gli attacchi di contraffazione sono generalmente impiegati da coloro che vogliono effettuare un'intrusione per evitare i controlli di sicurezza implementati dalla piattaforma del firewall.
- Infine, a causa del ridotto numero di variabili impiegate per prendere le decisioni nel controllo degli accessi, i firewall a filtraggio di pacchetti possono aprire inavvertitamente

mente brecce nella barriera di sicurezza dovute a configurazioni sbagliate. In altre parole, è facile configurare accidentalmente un firewall a filtraggio di pacchetti per consentire il passaggio di tipi di traffico di pacchetti, provenienti da sorgenti o diretti a destinazioni, che dovrebbero essere bloccati sulla base della politica di sicurezza informatica dell'organizzazione.

Ecco alcuni degli attacchi che possono essere perpetrati su un router a filtraggio di pacchetti e i relativi rimedi.

- **Contraffazione degli indirizzi IP (*IP spoofing*):** l'avversario, che vuole effettuare un'intrusione, trasmette un pacchetto dall'esterno, il cui campo relativo all'indirizzo IP sorgente contiene l'indirizzo di un host interno. L'attaccante spera che l'utilizzo di un indirizzo contraffatto gli consenta di penetrare quei sistemi che adottano misure di sicurezza molto semplici, relativamente all'indirizzo di sorgente, in base alle quali i pacchetti provenienti da specifici host interni fidati sono accettati. Il rimedio a tale tipo di attacco è scartare i pacchetti che provengono da un'interfaccia esterna avente un indirizzo sorgente interno.
- **Attacchi di instradamento di sorgente:** la stazione sorgente specifica l'instradamento che il pacchetto dovrebbe seguire quando attraversa Internet, nella speranza che questo eviterà le misure di sicurezza che non analizzano le informazioni di instradamento di sorgente. Il rimedio consiste nello scartare tutti i pacchetti che usano questa opzione.
- **Attacchi con piccoli frammenti:** colui che vuole effettuare un'intrusione utilizza l'opzione di frammentazione IP per creare frammenti molto piccoli. Ciò comporta che l'informazione TCP sarà contenuta in un frammento di pacchetto separato. Questo attacco è concepito per aggirare le regole di filtraggio che si basano sulle informazioni nell'intestazione TCP. Generalmente, un filtro di pacchetti prenderà la decisione se filtrare o no il pacchetto in base al primo frammento presente in quest'ultimo. Tutti i frammenti successivi di quel pacchetto verranno bloccati solamente basandosi sul fatto che sono parte del pacchetto il cui primo frammento è stato scartato. L'avversario spera che il router che effettua il filtraggio esamini solo il primo frammento, mentre tutti gli altri vengano lasciati passare senza ulteriori controlli. Un attacco tramite piccoli frammenti può essere evitato imponendo una regola che dica che il primo frammento di un pacchetto debba contenere una quantità predefinita minima dell'intestazione di trasporto. Se il primo frammento viene scartato, il filtro può ricordarsi del pacchetto, cui apparteneva quel frammento, e scartare tutti i frammenti successivi.

Firewall a controlli con memoria dello stato (*statefull inspection firewall*)

Un filtro di pacchetti tradizionale prende le decisioni sul filtraggio in base ai singoli pacchetti e non tiene in considerazione alcun contesto a livello più alto. Per capire cosa si intende per contesto, e perché un filtro di pacchetti tradizionale è limitato riguardo al contesto, è necessaria qualche piccola informazione. Molte delle applicazioni standardizzate, che girano al di sopra di TCP, seguono il modello client-server. Ad esempio, per il protocollo SMTP, la posta elettronica viene trasmessa da un sistema client a un sistema server. Il sistema client genera nuovi messaggi di posta elettronica tipicamente come input dell'utente. Il sistema server accetta i messaggi di posta elettronica in arrivo, e li mette nella casella di

Indirizzo sorgente	Porta sorgente	Indirizzo destinazione	Porta destinazione	Stato della connessione
192.168.1.100	1030	210.9.88.29	80	Stabilità
192.168.1.102	1031	216.32.42.123	80	Stabilità
192.168.1.101	1033	173.66.32.122	25	Stabilità
192.168.1.106	1035	177.231.32.12	79	Stabilità
223.43.21.231	1990	192.168.1.6	80	Stabilità
219.22.123.32	2112	192.168.1.6	80	Stabilità
210.99.212.18	3321	192.168.1.6	80	Stabilità
24.102.32.23	1025	192.168.1.6	80	Stabilità
223.212.212	1046	192.168.1.6	80	Stabilità

Tabella 11.2 Esempio di una tabella dello stato delle connessioni in un firewall con memoria dello stato.

posta dell'utente appropriato. SMTP funziona impostando una connessione TCP tra client e server nella quale il numero di porta TCP del server, che identifica l'applicazione server SMTP, è 25. Il numero di porta TCP per il client SMTP è un numero compreso tra 1024 e 65535, che viene generato dal client SMTP.

In generale, quando un'applicazione, che usa TCP, crea una sessione con un host remoto, crea una connessione TCP nella quale il numero di porta della applicazione server (remota) è un numero minore di 1024 e il numero di porta TCP dell'applicazione locale (client) è un numero tra 1024 e 65535. I numeri di porta minori di 1024 sono i numeri di porta "noti" e sono assegnati permanentemente a particolari applicazioni (ad esempio, 25 per il server SMTP). I numeri di porta tra 1024 e 65535 sono generati dinamicamente e hanno un significato temporaneo solo per la durata di una connessione TCP.

Un semplice firewall a filtraggio di pacchetti deve consentire il traffico di rete che torna indietro su tutte queste porte con numero elevato per consentire lo scambio del traffico basato su TCP. Ciò crea una vulnerabilità che può essere sfruttata dagli utenti non autorizzati.

Un filtro di pacchetti a controlli con memoria dello stato inasprisce le regole per il traffico TCP, creando un elenco delle connessioni TCP in uscita, come mostrato nella Tabella 11.2. Nell'elenco è presente un elemento per ciascuna connessione stabilita in un dato momento. Il filtro di pacchetti consentirà ora il traffico in ingresso verso una porta con numero elevato solo per quei pacchetti che rispondono al profilo di uno degli elementi in questo elenco.

Gateway a livello applicazione

Un gateway a livello applicazione, conosciuto anche come server proxy, agisce come un ripetitore per il traffico a livello applicazione (Figura 11.1b). L'utente contatta il gateway mediante un'applicazione TCP/IP quale, ad esempio, Telnet o FTP. Il gateway chiede all'utente il nome dell'host remoto cui vuole accedere. Quando l'utente risponde, fornendo un ID utente valido e informazioni di autenticazione, il gateway si mette in contatto con l'applicazione

sull'host remoto e trasmette i segmenti TCP contenenti i dati di livello applicazione tra le due parti.

Se il gateway non implementa il codice del proxy, per una specifica applicazione, il servizio non sarà supportato e non potrà essere infiltrato attraverso il firewall. Inoltre, il gateway può essere configurato per fornire supporto solo a specifiche funzionalità di un'applicazione, quelle cioè che l'amministratore di rete ritiene accettabili, mentre le altre funzionalità non sono consentite.

I gateway a livello applicazione sono in linea di massima più sicuri rispetto al filtraggio dei pacchetti. Invece di cercare di gestire a livello TCP e IP le molteplici possibili combinazioni che devono essere messe o vietate, il gateway a livello applicazione ha necessità di esaminare solo un numero ridotto di applicazioni ammissibili. Inoltre, è semplice effettuare operazioni di log e di audit a livello applicazione sul traffico in entrata.

Uno dei principali svantaggi di questo tipo di gateway è il carico addizionale di elaborazione che ogni connessione comporta. In realtà, ci sono due connessioni tra gli utenti finali in comunicazione, e il gateway – posto alla congiunzione delle due – deve pertanto esaminare e infiltrare tutto il traffico in entrambe le direzioni.

Gateway a livello circuito

Il gateway a livello circuito (Figura 11.1c) costituisce la terza tipologia di gateway. Questo tipo di gateway può essere realizzato mediante un sistema *stand-alone* o può essere una funzione specializzata realizzata per alcune applicazioni da un gateway a livello applicativo. Un gateway a livello circuito non consente connessioni TCP end-to-end; al contrario, il gateway stabilisce due connessioni TCP, una che connette il gateway a un utente TCP su un host interno e l'altra che connette il gateway a un utente TCP su un host esterno. Dopo aver stabilito le due connessioni, il gateway trasmette i segmenti TCP senza esaminarne il contenuto. La funzione di sicurezza consiste nell'individuare quali connessioni saranno messe.

Un tipico utilizzo dei gateway a livello circuito è quello in cui un amministratore di sistema si fida degli utenti interni. Il gateway può essere configurato per fornire un servizio a livello applicazione o di proxy sulle connessioni interne, e funzioni a livello circuito per connessioni esterne. Con questo tipo di configurazione, il gateway può essere sottoposto a un carico di elaborazione addizionale per l'analisi dei dati applicativi in arrivo relativi a funzioni non ammesse, ma non ha questo carico addizionale per quanto riguarda i dati in uscita.

Un esempio di implementazione di un gateway a livello circuito è costituito dal pacchetto SOCKS [KOB92]; la versione 5 di SOCKS è così definita nel documento RFC 1928.

Il protocollo qui descritto è stato progettato per fornire un'infrastruttura per applicazioni client-server in ambito TCP e UDP, che consenta loro di utilizzare in modo conveniente e sicuro i servizi di un firewall di rete. A livello concettuale, il protocollo si configura come un livello inserito tra il livello applicazione e il livello trasporto, e per tale motivo non fornisce i servizi di un gateway a livello rete come, ad esempio, l'inoltro di messaggi ICMP.

Il pacchetto SOCKS è formato dalle seguenti componenti.

- Il server SOCKS, che gira su un firewall UNIX.
- La libreria client SOCKS, che gira sugli host interni, protetti dal firewall.

- Versioni per SOCKS di alcuni programmi client standard come, ad esempio, FTP e Telnet. La realizzazione del protocollo SOCKS richiede generalmente la ricompilazione o il ricollegamento delle applicazioni client basate su TCP per fare in modo che siano utilizzate le procedure appropriate della libreria SOCKS.

Quando un client basato su TCP desidera stabilire una connessione con un'entità raggiungibile solo attraverso un firewall (tale verifica è lasciata all'implementazione), deve aprire un connessione TCP con la porta SOCKS appropriata del server SOCKS. Il servizio SOCKS è collocato sulla porta TCP 1080. Se la richiesta di connessione ha successo, il client inizia una negoziazione che riguarda il metodo di autenticazione da utilizzare, quindi si autentica attraverso il metodo concordato e invia una richiesta di collegamento. Il server SOCKS valuta la richiesta e decide se stabilire la connessione o negarla. Gli scambi UDP sono gestiti in maniera analoga: viene aperta una connessione TCP per autenticare l'utente in modo da consentirgli di inviare e ricevere segmenti UDP. I segmenti UDP sono infiltrati fino a quando la connessione TCP rimane aperta.

Bastion host

Un bastion host è un sistema che l'amministratore del firewall ritiene particolarmente critico per la sicurezza della rete. Generalmente, un bastion host è utilizzato come piattaforma per un gateway a livello applicazione o a livello circuito. Ecco alcune delle sue caratteristiche comuni.

- La piattaforma hardware del bastion host è dotata di una versione sicura del proprio sistema operativo, e ciò rende il sistema un sistema trusted.
- Sul bastion host sono installati solo quei servizi che l'amministratore di rete ritiene essenziali. Tale categoria di servizi include le applicazioni proxy come Telnet, DNS, FTP, SMTP e i servizi per l'autenticazione utente.
- Il bastion host può richiedere informazioni di autenticazione aggiuntive prima che all'utente sia consentito l'accesso ai servizi proxy. Inoltre, ogni servizio proxy può richiedere la propria forma di autenticazione prima di concedere l'accesso all'utente.
- Ciascun proxy è configurato in modo da fornire supporto solo per un sottoinsieme dell'insieme di comandi delle applicazioni standard.
- Ciascun proxy è configurato per consentire accessi solo a specifici sistemi host. Questo implica che l'insieme limitato di comandi a disposizione può essere applicato solo a un sottoinsieme di sistemi della rete che si sta proteggendo.
- Ciascun proxy mantiene informazioni di audit dettagliate tenendo traccia di tutto il traffico, di tutte le connessioni e della loro durata. Queste registrazioni sono uno strumento essenziale per scoprire e porre fine ad attacchi da parte di avversari.
- Ciascun modulo proxy è un pacchetto software di dimensioni molto ridotte, specificatamente progettato per la sicurezza di rete. La relativa semplicità di tali moduli rende più semplice verificare l'assenza di punti deboli rispetto alla sicurezza. Ad esempio, una tipica applicazione UNIX per la gestione della posta elettronica può essere costituita da più di 20.000 linee di codice, mentre un proxy per la gestione della posta può essere composto da meno di 1.000 linee di codice.

- Ciascun proxy è indipendente dagli altri che agiscono sul bastion host. Se si rilevano problemi con l'operato di un qualsiasi proxy, o se viene rilevata una sua possibile vulnerabilità, questo può essere disinstallato senza che ciò influenzi l'operato delle altre applicazioni proxy. Inoltre, nel caso in cui gli utenti richiedano il supporto di un nuovo tipo di servizio, l'amministratore di rete può agevolmente installare sul bastion host il corrispondente proxy.
- Un proxy non effettua generalmente accessi a disco, ad eccezione di quelli necessari per leggere il suo file di configurazione iniziale. Questa caratteristica rende difficile l'installazione di "cavalli di Troia" o altri file pericolosi sul bastion host.
- Ciascun proxy viene eseguito come utente non privilegiato in una directory privata e sicura del bastion host.

Configurazione dei firewall

Oltre all'utilizzo di configurazioni semplici, che consistono in un sistema unico come, ad esempio, un singolo router a filtraggio di pacchetti o un singolo gateway (Figura 11.1), sono possibili configurazioni più complesse, più comunemente utilizzate. La Figura 11.2 illustra tre comuni configurazioni dei firewall.

Nella configurazione **screened host firewall, single-homed bastion**, cioè con bastion host avente una sola connessione di rete (Figura 11.2a), il firewall è composto da due sistemi: un router a filtraggio di pacchetti e un bastion host. Tipicamente, il router è configurato in modo che:

- per quanto riguarda il traffico in arrivo da Internet, sono accettati in ingresso solo i pacchetti IP destinati al bastion host;
- per il traffico che ha origine dalla rete interna, è concessa l'uscita solo ai pacchetti IP provenienti dal bastion host.

Il bastion host effettua funzioni di autenticazione e di proxy. Questa configurazione presenta maggiori caratteristiche di sicurezza di quella composta da un unico router a filtraggio di pacchetti o da un unico gateway a livello applicazione. Le ragioni sono essenzialmente due. In primo luogo, questa configurazione realizza sia il filtraggio a livello pacchetto sia quello a livello applicazione, consentendo notevole flessibilità nella definizione della politica di sicurezza da adottare. In secondo luogo, chi vuole effettuare un'intrusione deve – in linea generale – penetrare all'interno di due sistemi distinti prima di compromettere la sicurezza della rete interna.

Questa configurazione offre, inoltre, una certa flessibilità nel fornire accessi diretti a Internet. Ad esempio, la rete interna può contenere un server con informazioni pubbliche – quale, ad esempio, un server web – per cui non è necessario un livello di sicurezza elevato. In questo caso, il router può essere configurato in modo da consentire traffico diretto tra il server e Internet.

Nella configurazione **single-homed** appena descritta, se il router a filtraggio di pacchetti viene completamente compromesso, il traffico da Internet agli host sulla rete interna potrebbe attraversare il router senza che venga effettuato alcun controllo. La configurazione **screened host firewall, dual-homed bastion** (Figura 11.2b), cioè con un host bastion avente due con-

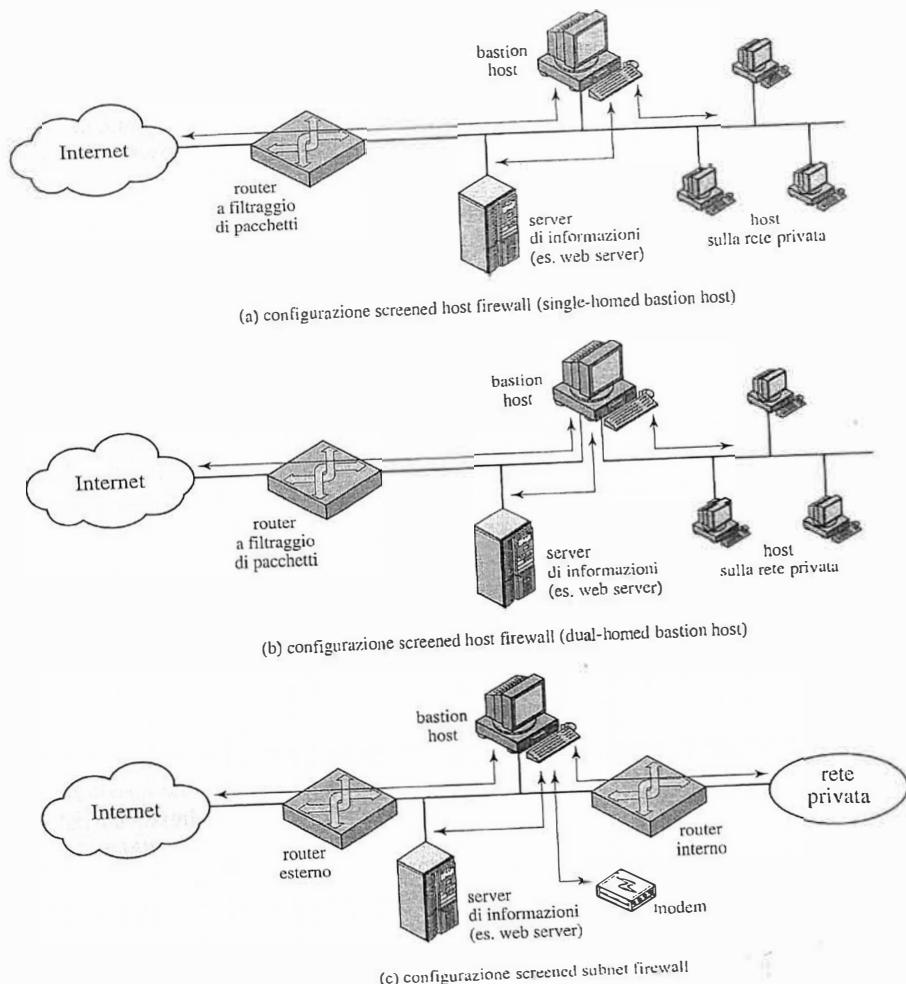


Figura 11.2 Configurazioni dei firewall.

nessioni di rete, previene, a livello fisico, questo tipo di violazione di sicurezza. Anche in questa configurazione abbiamo gli stessi vantaggi presenti nella precedente configurazione, relativi alla presenza di due livelli di sicurezza. Inoltre, anche in questa configurazione, a un router, se ciò è in accordo con la politica di sicurezza in vigore.

La configurazione **screened subnet firewall** (Figura 11.2c) è la più sicura tra quelle considerate. In questa configurazione si utilizzano due router a filtraggio di pacchetti, uno

collocato tra il bastion host e Internet e l'altro tra il bastion host e la rete interna. Mediante questa configurazione si crea una sottorete isolata, che può semplicemente comprendere solo il bastion oppure può contenere uno o più server e modem. Tipicamente, sia la rete Internet che quella interna hanno accesso agli host posti sulla sottorete protetta, ma è bloccato il traffico attraverso la sottorete protetta. Questo tipo di configurazione offre alcuni vantaggi.

- Prevede tre livelli di difesa per contrastare le intrusioni.
- Il router esterno segnala alla rete Internet solo la presenza della sottorete protetta; la rete interna non è quindi visibile da Internet.
- Analogamente, il router interno segnala alla rete interna solo l'esistenza della sottorete protetta; questo implica che i sistemi posti sulla rete interna non possono stabilire connessioni dirette a Internet.

11.2 Sistemi trusted

La capacità di un sistema di difendersi da intrusioni e programmi dolosi può essere accresciuta utilizzando la tecnologia dei sistemi trusted. Questo paragrafo fornisce una breve panoramica sull'argomento, iniziando da alcuni concetti base di controllo dell'accesso ai dati.

Controllo dell'accesso ai dati

Dopo che un utente si è collegato, ottiene l'accesso a uno o più insiemi di host ed applicazioni. Questo tipo di controllo non è di solito sufficiente per sistemi le cui basi di dati contengono dati sensibili. Tramite la procedura di controllo dell'accesso, un utente può essere identificato dal sistema. A ciascun utente può essere associato un profilo che specifica operazioni e accessi ai file a lui consentiti. Il sistema operativo può, quindi, impostare regole di accesso sulla base dei profili utente. Il sistema di gestione di basi di dati deve comunque controllare l'accesso a specifici record o anche a porzioni di questi. Ad esempio, può essere consentito a ciascun membro dell'amministrazione di ottenere la lista del personale della società, mentre solo particolari persone possono avere accesso alle informazioni relative agli stipendi. Se il sistema operativo può concedere a un utente il permesso di accedere a un file o utilizzare un'applicazione – ed a questo punto non effettuare più alcun controllo di sicurezza – il sistema di gestione di basi di dati deve controllare ogni singolo tentativo di accesso. La decisione di accordare o meno l'accesso dipenderà non solo dall'identità dell'utente che richiede l'accesso, ma anche dalla specifica porzione di dati cui l'utente vuole accedere ed dalle informazioni cui tale utente ha avuto in precedenza accesso.

Un modello generale di controllo dell'accesso, per sistemi di gestione di file o di basi di dati, è quello conosciuto come **matrice d'accesso** (Figura 11.3a), caratterizzato dai seguenti elementi costitutivi.

- **Soggetto.** Ogni entità che può richiedere accessi agli oggetti. Generalmente, il concetto di soggetto è sinonimo di processo, in quanto ciascun utente o applicazione ottiene in realtà un accesso a un oggetto tramite un processo.

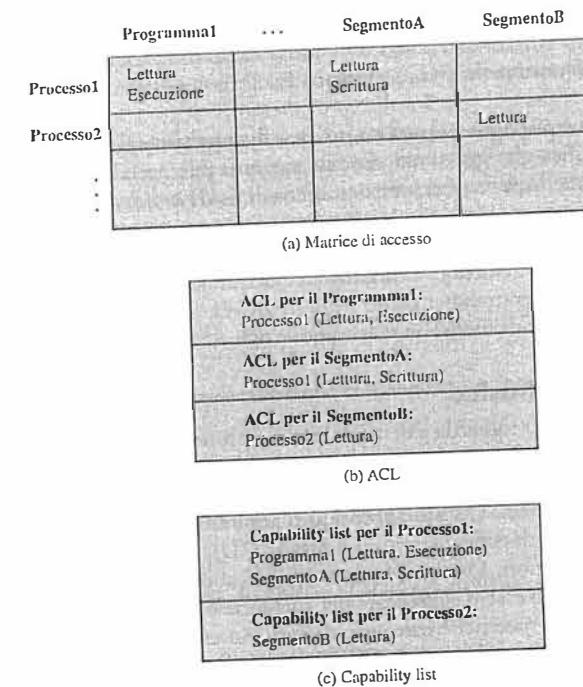


Figura 11.3 Strutture di controllo dell'accesso.

- **Oggetto.** Ogni entità il cui accesso deve essere controllato. Esempi di oggetti includono file o loro parti, programmi e segmenti di memoria.
- **Diritto di accesso.** Modalità di accesso che un soggetto può esercitare su un certo oggetto. Esempi di diritti di accesso sono il diritto di lettura, scrittura o esecuzione.

Una delle dimensioni della matrice di accesso rappresenta i soggetti che possono richiedere l'accesso ai dati. Generalmente, la lista di soggetti è costituita da singoli utenti o gruppi, sebbene il controllo dell'accesso possa essere effettuato per terminali, host o applicazioni in sostituzione o in aggiunta al controllo dell'accesso a livello utente. L'altra dimensione della matrice rappresenta gli oggetti cui è possibile accedere. Al livello di granularità più fine, gli oggetti possono essere singoli campi dati. Gli oggetti possono anche essere a un livello di aggregazione maggiore come, ad esempio: record, file o, anche, un'intera base di dati. Ogni elemento della matrice stabilisce i diritti di accesso del corrispondente soggetto sull'oggetto considerato.

In pratica, la matrice d'accesso è di solito sparsa ed è quindi realizzata mediante una sua scomposizione rispetto a una delle dimensioni. Scomponendo la matrice per colonne, si ottiene un insieme di liste, chiamate **access control list** (ACL) illustrate nella Figura 11.3b. Un'ACL elenca, per ciascun oggetto, gli utenti che hanno il diritto di accedervi e le modalità

di accesso che tali utenti possono esercitare su tale oggetto. L'ACI può contenere una voce di default, o pubblica, che consente di assegnare un insieme di diritti di default a utenti che non sono esplicitamente elencati nella lista. Gli elementi della lista possono essere sia singoli utenti sia gruppi di utenti.

La scomposizione per righe origina un insieme di **capability ticket** (Figura 11.3c). Un capability ticket specifica gli oggetti cui ciascun soggetto può accedere e i relativi diritti di accesso. Ciascun utente dispone di un certo numero di ticket e può essere autorizzato a prestarli o cederli ad altri utenti. Dato che i ticket possono essere disseminati all'interno del sistema, presentano un problema di sicurezza aggiuntivo rispetto alle ACL, cioè si deve assicurare che il ticket non possa essere contraffatto. Un metodo per ottenere questo è che i ticket siano gestiti dal sistema operativo per conto degli utenti. In questo caso, i ticket dovrebbero essere conservati in un'area di memoria inaccessibile agli utenti.

Concetto di sistema trusted

Quanto fino ad ora esposto riguarda essenzialmente la protezione di messaggi o dati da attacchi attivi e passivi perpetrati da specifici utenti. Un'esigenza, per certi aspetti diversa ma comunque applicabile a un ampio numero di contesti, è la protezione dei dati e delle risorse sulla base di livelli di sicurezza. Questo approccio si applica comunemente ai contesti militari dove l'informazione è classificata come *unclassified* (U), *confidential* (C), *secret* (S), *top secret* (TS), o a livelli maggiori. Questo concetto è applicabile anche a tutti quei domini in cui le informazioni possono essere classificate per categorie e agli utenti possono essere concessi permessi di accesso a specifiche categorie di dati. Ad esempio, il più elevato livello di sicurezza può essere attribuito a documenti e dati relativi alla pianificazione strategica aziendale, che devono essere accessibili solo da funzionari aziendali e dal loro staff; a un livello immediatamente inferiore potrebbero essere classificati dati sensibili di tipo finanziario o relativi al personale, accessibili solo da soggetti quali il personale amministrativo e i funzionari aziendali, e così via.

Se sono definite molteplici categorie e livelli di dati, si parla di **sicurezza multilivello**. Il principio base della sicurezza multilivello è che un soggetto a un livello alto non può trasferire informazioni a un soggetto di livello più basso o non compatibile, salvo la specifica volontà di un utente autorizzato. Per scopi di implementazione, questo requisito è suddiviso in due parti, con una semplice formulazione. Un sistema che realizza la sicurezza multilivello deve rispettare i seguenti principi.

- **Nessuna lettura verso l'alto (*no read up*)**. Un soggetto può leggere solo oggetti con livello di sicurezza uguale o inferiore al suo. Tale proprietà è conosciuta in letteratura come **simple security property** (proprietà di sicurezza semplice).
- **Nessuna scrittura verso il basso (*no write down*)**. Un soggetto può scrivere solo in oggetti con un livello di sicurezza maggiore o uguale al suo. Tale proprietà è conosciuta in letteratura come ***-property** (proprietà *, pronunciata *star property*)¹.

Tali regole, se fatte rispettare nel modo appropriato, forniscono la sicurezza multilivello. Nel caso di un sistema di elaborazione dati, l'approccio comunemente adottato – oggetto di molta attività di ricerca e sviluppo – è basato sul concetto di **reference monitor**. Questo approccio è illustrato nella Figura 11.4. Il reference monitor è un dispositivo di controllo realizzato a livello hardware e di sistema operativo, che regolamenta gli accessi dei soggetti agli oggetti sulla base dei parametri di sicurezza del soggetto e dell'oggetto in questione. Il reference monitor ha accesso a un file, chiamato **security kernel database**, che elenca i privilegi di accesso di ciascun soggetto (autorizzazione di sicurezza) e gli attributi di protezione di ciascun oggetto (livello di classificazione). Il reference monitor ha lo scopo di far rispettare le regole di sicurezza (*no read up, no write down*) e possiede le seguenti proprietà.

- **Mediazione totale**. Le regole di sicurezza sono applicate a ciascun accesso, non soltanto quindi, ad esempio, alle operazioni di apertura di un file.
- **Isolamento**. Sia il reference monitor che la base di dati sono protette da modifiche non autorizzate.
- **Verificabilità**. Deve essere possibile provare la correttezza del reference monitor. Questo implica che deve essere possibile dimostrare matematicamente che il reference monitor realizza le regole di sicurezza e fornisce mediazione totale e isolamento.

I requisiti sopra esposti sono molto rigidi. Il requisito relativo alla mediazione totale implica che ciascun accesso ai dati, sia in memoria centrale sia su disco o nastro, debba essere me-

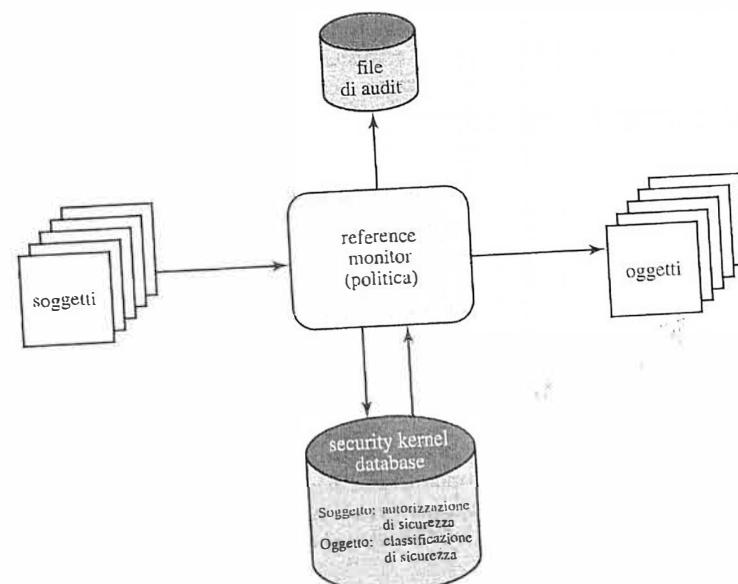


Figura 11.4 Reference monitor.

¹ Internet control message protocol (ICMP) è un protocollo a livello IP per lo scambio di pacchetti di controllo tra un router e un host o tra host. Il pacchetto ECHO richiede al ricevente di rispondere con una risposta all'ECHO per verificare se tra le due entità sia possibile la comunicazione.

diato. Le implementazioni totalmente software non sono abbastanza efficienti per essere utilizzate in pratica; la soluzione è una realizzazione hardware almeno parziale. Il requisito di isolamento comporta che non deve essere possibile per un avversario, indipendentemente dalla sua abilità, modificare la logica del reference monitor o il contenuto del security kernel database. Infine, il requisito di verificabilità matematica è di difficilissima realizzazione per una realtà complessa come un calcolatore di uso generale. Un sistema che può soddisfare il requisito di verificabilità viene chiamato **sistema trusted**.

L'ultimo elemento illustrato nella Figura 11.4 è un file di audit. Il file di audit memorizza eventi rilevanti dal punto di vista della sicurezza, quali violazioni di sicurezza scoperte o modifiche autorizzate al security kernel database.

Il Dipartimento della Difesa degli Stati Uniti, per soddisfare le proprie necessità e come servizio pubblico, ha istituito nel 1981 il Computer Security Center all'interno della National Security Agency (NSA), con lo scopo di favorire la più ampia diffusione dei sistemi trusted di elaborazione. Questo obiettivo si concretizza tramite il Commercial Product Evaluation Program (programma di valutazione dei prodotti commerciali). In breve, il centro si prefigge di valutare in quale misura i prodotti disponibili in commercio soddisfano i requisiti di sicurezza precedentemente descritti. Il centro stabilisce una classifica dei prodotti, valutati in base alle funzionalità di sicurezza che essi forniscono. Tali valutazioni sono necessarie per gli scopi del Dipartimento della Difesa ma sono anche pubblicate e disponibili liberamente. Possono quindi servire come linee guida per gli acquirenti di prodotti commerciali.

Difesa dai cavalli di Troia

Un metodo per difendersi dagli attacchi perpetrati mediante cavalli di Troia (*Trojan horse*), è l'utilizzo di un sistema operativo sicuro e trusted. La Figura 11.5 illustra un esempio in cui un cavallo di Troia è utilizzato per aggirare i controlli effettuati tramite ACL, che rappresenta il meccanismo di sicurezza standard, utilizzato dalla maggior parte dei sistemi operativi e dei sistemi di gestione dati. Nell'esempio, un utente chiamato Bob interagisce, tramite un programma, con un file di dati contenente la stringa di caratteri "CPE170KS", che rappresenta un'informazione sensibile. Bob ha creato il file ed ha concesso il privilegio di lettura e scrittura solo a programmi eseguiti per suo conto. Ciò implica che solo processi di cui Bob è proprietario possono accedere al file.

L'attacco mediante un cavallo di Troia inizia quando un utente ostile, di nome Alice, ottiene un accesso legittimo al sistema e installa su di esso sia un programma che funge da cavallo di Troia sia un file che viene utilizzato durante l'attacco alla stregua di una "tasca posteriore". Alice concede a se stessa il diritto di lettura e scrittura su tale file, mentre concede a Bob il diritto di sola scrittura (Figura 11.5a). A questo punto, persuade Bob a invocare il programma che agisce da cavallo di Troia, ad esempio spacciandolo come una funzione di utilità. Quando il programma rileva di essere eseguito per conto di Bob, legge la stringa di caratteri sensibile dal file di Bob e la copia nel file di Alice (Figura 11.5b). Sia l'operazione di lettura che quella di scrittura soddisfano i vincoli specificati dalle ACL. A questo punto Alice deve solo accedere in un momento successivo al file, per entrare in possesso del valore della stringa.

Si consideri ora l'utilizzo di un sistema operativo sicuro in questo scenario (Figura 11.5c). Al momento della connessione, ai soggetti viene assegnato un livello di sicurezza, sulla base di criteri quali il terminale da cui si richiede l'accesso e l'utente coinvolto nella richiesta, identi-

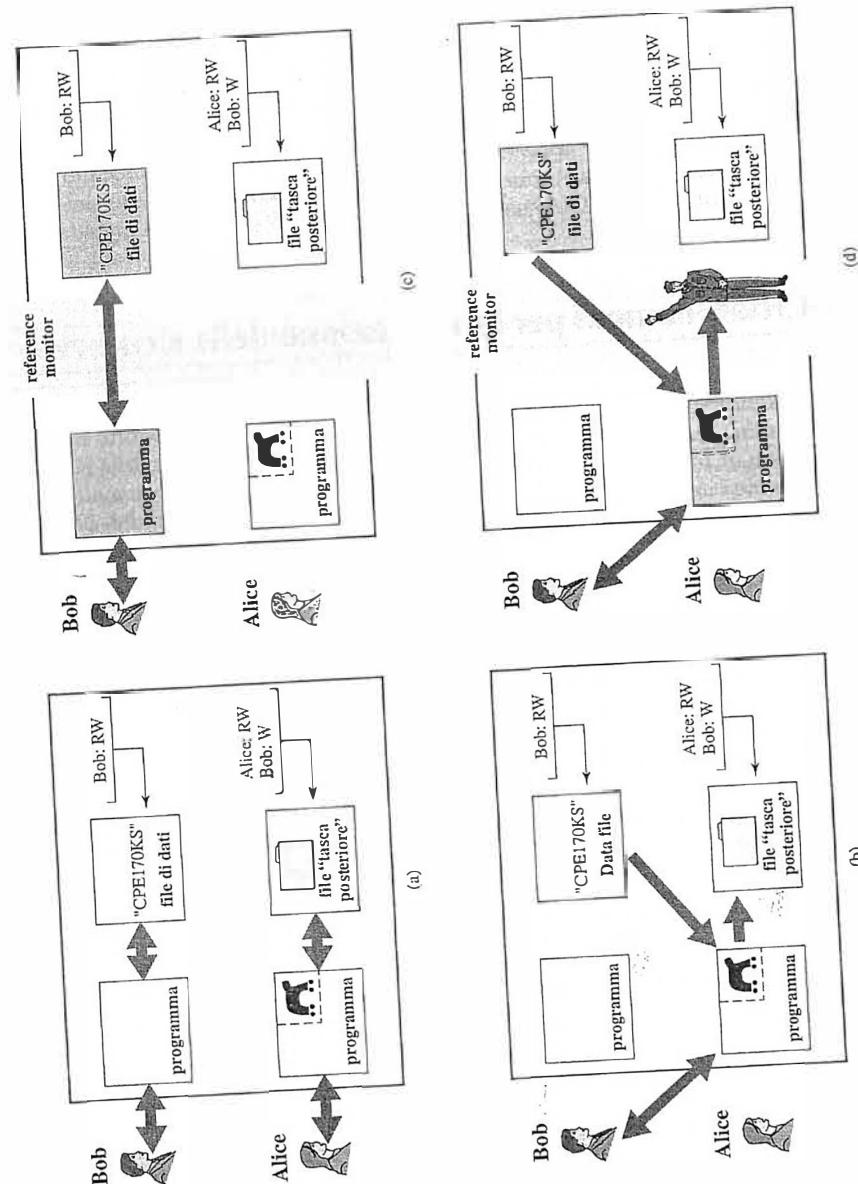


Figura 11.5 Cavallo di Troia e sistema operativo sicuro.

ficato mediante ID e password. Nell'esempio considerato, esistono due livelli di sicurezza, sensibile e pubblico, ordinati in modo che il primo sia superiore al secondo. Ai processi posseduti da Bob ed ai suoi file dati viene assegnato il livello di sicurezza sensibile, mentre i processi e i file di Alice sono considerati a livello pubblico. Se Bob invoca il programma che opera come cavallo di Troia (Figura 11.5d), tale programma acquisisce il livello di sicurezza di Bob. Può quindi, in base alla simple security property, leggere la stringa di caratteri sensibile. Questo comporta una violazione della *-Property e quindi il tentativo viene impedito dal reference monitor. Ciò implica che il tentativo di scrivere nel file di Alice fallisce anche se questa scrittura era permessa dall'ACL, questo perché la politica di sicurezza ha precedenza rispetto al meccanismo delle ACL.

11.3 Criteri comuni per la valutazione della sicurezza

Il lavoro, fatto dal National Security Agency e da altre agenzie governative statunitensi, di sviluppare i requisiti e i criteri di valutazione per i sistemi fidati è stato riprodotto in lavori simili in altri paesi. I **criteri comuni** (CC, *common criteria*) per la valutazione della sicurezza informatica sono un'iniziativa internazionale dei gruppi di standardizzazione in molti paesi, per sviluppare uno standard internazionale che consenta di specificare i requisiti di sicurezza e definirne i criteri di valutazione.

Requisiti

I CC definiscono un insieme comune di potenziali requisiti di sicurezza da usare nella valutazione. Il termine **target of evaluation** (TOE, oggetto della valutazione) si riferisce a quella parte di prodotto o sistema che è soggetto alla valutazione. I requisiti ricadono in due categorie.

- **Requisiti funzionali:** definiscono il comportamento di sicurezza desiderato. I documenti CC stabiliscono un insieme di componenti funzionali di sicurezza, che forniscano un modo standard per esprimere i requisiti funzionali di sicurezza di un TOE.
- **Requisiti di garanzia:** consentono di accettare se le misure di sicurezza dichiarate sono efficaci e implementate correttamente. I documenti CC stabiliscono un insieme di componenti di garanzia che forniscono un modo standard per esprimere i requisiti di garanzia per un TOE.

Entrambi i requisiti funzionali e di garanzia sono organizzati in classi: una **classe** è un insieme di requisiti che condividono un obiettivo o uno scopo comuni.

Le Tabelle 11.3 e 11.4 definiscono brevemente le classi dei requisiti funzionali e di garanzia. Ciascuna di queste classi contiene molte famiglie. I requisiti all'interno di ciascuna famiglia condividono gli obiettivi di sicurezza, ma differiscono sull'enfasi o sulla rigorosità. Ad esempio, la classe di audit contiene sei famiglie che hanno a che fare con vari aspetti dell'audit (ad esempio: generazione dei dati di audit, analisi dell'audit, memorizzazione degli eventi dell'audit). Ciascuna famiglia, a sua volta, contiene uno o più componenti. Un **componente** è un insieme di requisiti di sicurezza per l'inserimento nelle strutture definite nei CC.

Classe	Descrizione
Audit	Coinvolge il riconoscimento, la registrazione, la memorizzazione e l'analisi delle informazioni legate alle attività di sicurezza. La documentazione di audit viene prodotta da queste attività e può essere esaminata per determinarne l'importanza nella sicurezza.
Supporto Crittografico	Usato quando il TOE implementa funzioni crittografiche. Queste possono essere usate, ad esempio, per: supportare le comunicazioni, l'identificazione o l'autenticazione o la separazione dei dati.
Comunicazioni	Fornisce due famiglie che riguardano il non ripudio dei dati da parte della sorgente o del ricevitore dei dati.
Protezione dei dati utente	Specifica i requisiti legati alla protezione dei dati utente all'interno del TOE durante l'importazione, l'esportazione e la memorizzazione, oltre agli attributi di sicurezza legati ai dati utente.
Identificazione e autenticazione	Assicura l'identificazione non ambigua degli utenti autorizzati e la corretta associazione degli attributi di sicurezza con utenti e oggetti.
Gestione della sicurezza	Specifica come vengono gestiti gli attributi di sicurezza, i dati e le funzioni.
Privacy	Offre all'utente la protezione dalla scoperta e dall'uso scorretto della sua identità da parte di altri utenti.
Protezione delle funzioni di sicurezza di un TOE	È focalizzato sulla protezione dei dati di TSF (<i>TOE security function</i> , funzioni di sicurezza di un TOE), piuttosto che dei dati utente. La classe concerne l'integrità e la gestione dei meccanismi e dei dati di TSF.
Utilizzo delle risorse	Supporta la disponibilità delle risorse richieste, come la capacità di elaborazione e di memorizzazione. Comprende i requisiti per la tolleranza ai guasti, la priorità del servizio e l'allocazione delle risorse.
Accesso al TOE	Specifica dei requisiti funzionali, oltre a quelli specificati per l'identificazione e l'autenticazione, per controllare la costituzione di una sessione utente. I requisiti per l'accesso al TOE amministrano elementi come la limitazione nel numero e nell'ambito delle sessioni utente, la visualizzazione della storia degli accessi e la modifica ai parametri di accesso.
Porte/Canali fidati	Riguardano i canali di comunicazione fidati tra utenti e TSF e tra TSF.

Tabella 11.3 Requisiti funzionali di sicurezza dei CC.

Ad esempio, la classe di supporto crittografico di requisiti funzionali comprende due famiglie: la gestione delle chiavi crittografiche e le operazioni crittografiche. Ci sono quattro componenti nella famiglia della gestione delle chiavi crittografiche che vengono usate per specificare: l'algoritmo di generazione e la dimensione della chiave, il metodo di distribuzione della chiave, il metodo di accesso della chiave e il metodo di distruzione della chiave. Per ciascuna componente si può fare riferimento a uno standard per definire il requisito. Nella famiglia delle operazioni crittografiche vi è un solo componente che specifica un algoritmo e la dimensione della chiave, basandosi su uno standard assegnato.

Gli insiemi di componenti funzionali e di garanzia possono essere raggruppati in un insieme di pacchetti riusabili, noti per essere utili nel raggiungimento di obiettivi identificati. Un esempio di questi pacchetti potrebbero essere i componenti funzionali necessari per i **controlli discrezionali degli accessi** (*discretionary access control*).

Classe	Descrizione
Gestione delle configurazioni	Richiede che sia adeguatamente preservata l'integrità del TOE. Specificatamente, la gestione delle configurazioni fornisce la sicurezza che il TOE e la documentazione usata per la valutazione sono quelle preparate per la distribuzione.
Distribuzione ed esercizio	Riguarda le misure, le procedure e gli standard per la distribuzione, installazione e l'uso in esercizio sicuri del TOE, per assicurare che la protezione di sicurezza offerta dal TOE non sia compromessa durante questi eventi.
Sviluppo	Riguarda il miglioramento del TSF dalle specifiche definite nella ST per l'implementazione e la mappatura dei requisiti di sicurezza nella rappresentazione a livello più basso.
Documenti di linee guida	Riguarda l'uso in esercizio sicuro del TOE, da parte degli utenti e degli amministratori.
Supporto al ciclo di vita	Riguarda il ciclo di vita del TOE e comprende la definizione del ciclo di vita, gli strumenti e le tecniche, la sicurezza dell'ambiente di sviluppo e i provvedimenti trovati dai fruitori del TOE.
Test	Riguarda la dimostrazione che il TOE verifica i propri requisiti funzionali. Le famiglie hanno come fine la copertura e la profondità dei test durante lo sviluppo e i requisiti per i test indipendenti.
Valutazione della vulnerabilità	Definisce i requisiti indirizzati all'identificazione di vulnerabilità sfruttabili, che potrebbero essere introdotte durante la realizzazione, l'esercizio, l'uso improprio o la configurazione scorretta del TOE. Le famiglie qui identificate riguardano l'identificazione delle vulnerabilità attraverso l'analisi di canali nascosti, l'analisi della configurazione del TOE, l'esame della robustezza dei meccanismi delle funzioni di sicurezza e l'identificazione di errori introdotti durante lo sviluppo del TOE. La seconda famiglia tratta la categorizzazione di sicurezza degli elementi del TOE. La terza e la quarta trattano l'analisi dei cambiamenti per un impatto di sicurezza e la fornitura delle prove che quelle procedure sono state seguite. Questa classe fornisce i blocchi di costruzione per la costituzione di schemi per il mantenimento delle garanzie di sicurezza.
Mantenimento della garanzia	Fornisce i requisiti che sono designati per essere applicati dopo che un TOE è stato certificato a fronte dei CC. Questi requisiti hanno lo scopo di assicurare che il TOE continuerà a soddisfare i propri obiettivi di sicurezza quando verranno fatti dei cambiamenti al TOE o al suo ambiente.

Tabella 11.4 Requisiti di garanzia di sicurezza dei CC

Profili e obiettivi

I CC definiscono anche due tipi di documenti che possono essere generati usando i requisiti definiti nei CC.

- **Profili di protezione (PP, protection profile):** definiscono un insieme di requisiti e obiettivi di sicurezza indipendenti dall'implementazione per una categoria di prodotti o sistemi che soddisfano necessità simili nella sicurezza IT degli utenti. Un PP si propone per essere riusabile e per definire requisiti noti per essere utili ed efficaci nel soddisfare gli obiettivi identificati. Il concetto dei PP è stato sviluppato per supportare la

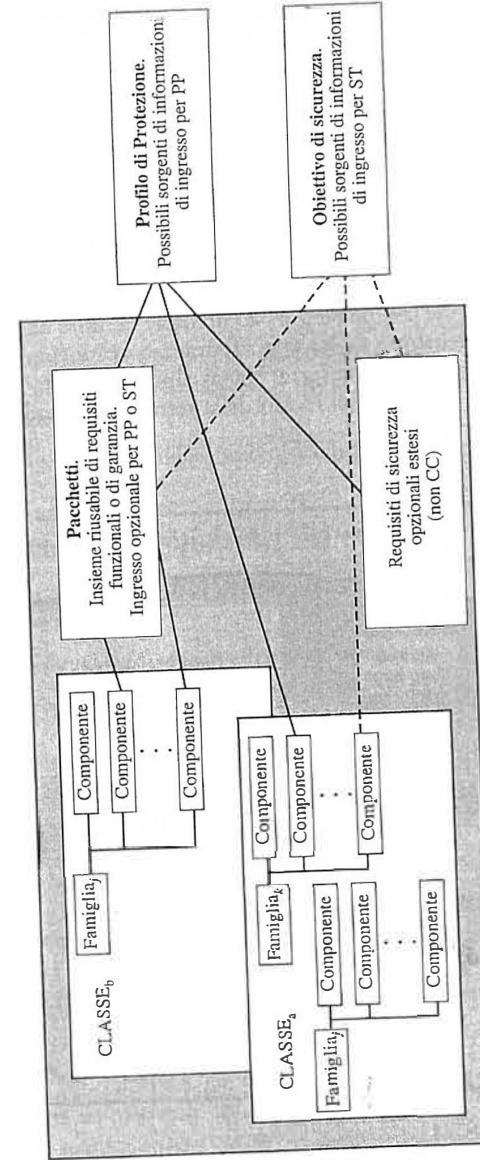


Figura 11.6 Organizzazione e costruzione dei requisiti dei criteri comuni.

definizione di standard funzionali e come aiuto nella definizione di specifiche di approssimazione. I PP riflettono i requisiti di sicurezza degli utenti.

- **Obiettivi di sicurezza (ST, security target):** contengono gli obiettivi e i requisiti di sicurezza IT di uno specifico TOE e definiscono le misure funzionali e di garanzia offerte da quel TOE per soddisfare i requisiti dichiarati. Gli ST possono dichiarare la conformità a uno o più PP e formano le basi per una valutazione. Gli ST sono offerti dai fornitori o dagli sviluppatori.

Nella Figura 11.6, a pagina precedente, sono rappresentate le relazioni tra i requisiti (da una parte) e i profili e gli obiettivi (dall'altra). Per un PP, un utente può selezionare molte componenti per definire i requisiti del prodotto desiderato. L'utente può anche far riferimento a pacchetti predefiniti che assegnano parecchi requisiti, comunemente raggruppati all'interno di un documento di requisiti di un prodotto. In modo simile, un fornitore o un progettista può selezionare molte componenti e pacchetti per definire un ST.

Nella Figura 11.7 è mostrato a cosa si fa riferimento nei documenti di CC come paradigma dei requisiti funzionali di sicurezza. Essenzialmente, questa illustrazione è basata sul concetto di reference monitor, ma fa uso della terminologia e della filosofia di protezione dei CC.

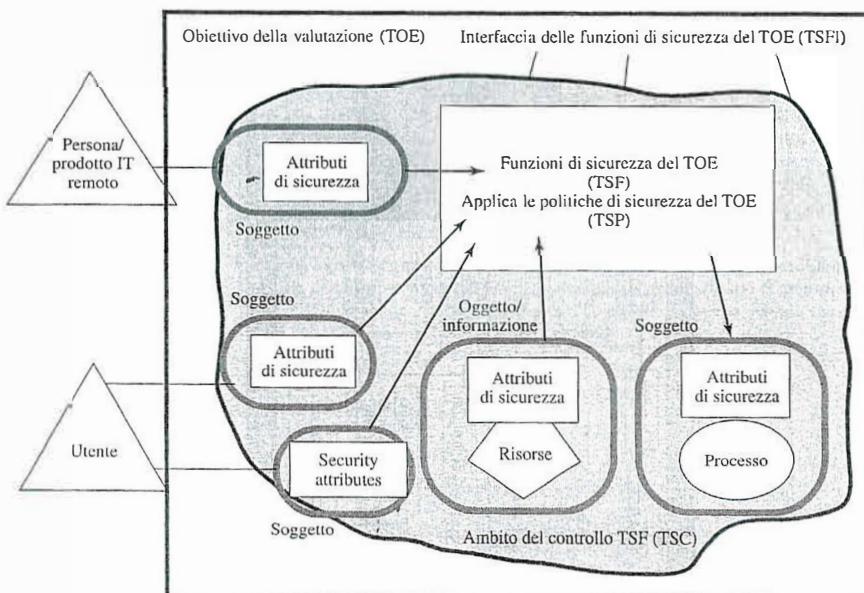


Figura 11.7 Paradigma dei requisiti funzionali di sicurezza.

11.4 Letture consigliate e siti web

[CHAP00] fornisce una trattazione classica dei firewall. Un altro classico, aggiornato di recente, è [CHES03]. [LODI98], [OPPL97] e [BELL94b] sono valide rassegne sull'argomento. [WACK02] costituisce un'eccellente panoramica delle tecnologie e delle politiche dei firewall. [AUDI04] e [WILS05] forniscono utili trattazioni sui firewall.

[GASS88] fornisce uno studio completo sui sistemi di elaborazione trusted. [PFLE97] e [GOLL99] rappresentano un'ulteriore valida fonte di informazioni. [FELT03] e [OPPL05] rappresentano un'utile trattazione sui sistemi di elaborazione trusted.

Siti web consigliati

- **Firewall.com:** numerosi collegamenti a riferimenti sui firewall e risorse software.
- **Trusted Computing Group:** gruppo di fornitori coinvolto nello sviluppo e nella promozione standard sui computer fidati. Il sito comprende articoli, specifiche e collegamenti ai fornitori.
- **Common Criteria Portal:** il sito web ufficiale del progetto dei criteri comuni.

11.5 Domande di revisione ed esercizi

Domande di revisione

- 11.1 Elenca i tre obiettivi per il progetto di un firewall.
- 11.2 Elenca quattro tecniche usate dai firewall per controllare gli accessi e applicare una politica di sicurezza.
- 11.3 Quali informazioni vengono usate da un tipico router a filtraggio dei pacchetti?
- 11.4 Quali sono alcuni dei punti deboli di un router a filtraggio dei pacchetti?
- 11.5 Qual è la differenza tra un router a filtraggio dei pacchetti e un firewall a controllo con memoria dello stato?
- 11.6 Che cos'è un gateway a livello applicazione?
- 11.7 Che cos'è un gateway a livello di circuito?
- 11.8 Quali sono le differenze tra le tre configurazioni della Figura 11.2?
- 11.9 Nel contesto del controllo degli accessi, qual è la differenza tra un soggetto e un oggetto?
- 11.10 Qual è la differenza tra un access control list e un capability ticket?
- 11.11 Quali sono le due regole che applica un reference monitor?
- 11.12 Quali proprietà sono richieste da un reference monitor?
- 11.13 Quali sono i criteri comuni?

Esercizi

- 11.1 Come è stato ricordato nel Paragrafo 11.1, un approccio per annullare un attacco con piccoli frammenti è quello di far rispettare la regola che il primo frammento di un pacchetto IP deve contenere una lunghezza minima dell'intestazione di trasporto. Se il primo frammento viene bloccato, tutti i successivi frammenti possono essere bloccati. Tuttavia, le natura di IP è tale che i frammenti possono arrivare fuori ordine. Quindi, un frammento intermedio può passare attraverso il filtro prima che il frammento iniziale venga bloccato. Come può essere gestita questa situazione?
- 11.2 In un pacchetto IPv4, la dimensione del payload nel primo frammento, in byte, è pari alla Lunghezza totale – (4 × HIL). Se questo valore è meno del minimo richiesto (8 byte per TCP), allora questo frammento e l'intero pacchetto verrà bloccato. Suggerite metodi alternativi per raggiungere lo stesso risultato usando solo il campo Spostamento del frammento.
- 11.3 La RFC 791, la specifica del protocollo IPv4, descrive un algoritmo per il riassemblaggio che da luogo alla riscrittura di nuovi frammenti sulle porzioni che si sovrappongono dei frammenti precedentemente ricevuti. Data un'implementazione del riassemblaggio di questo tipo, l'attaccante potrebbe costruire una serie di pacchetti nei quali il frammento più basso (offset zero) contiene dati innocui (e pertanto viene fatto passare dai filtri amministrativi di pacchetti), mentre alcuni dei pacchetti successivi, aventi offset non nullo, si sovrappongono in parte alle informazioni dell'intestazione TCP (porta di destinazione, per esempio) e ne causano la modifica. Il secondo pacchetto passerebbe attraverso la maggior parte delle implementazioni dei filtri, perché non ha lo spostamento zero del frammento. Si suggerisce un metodo che potrebbe essere usato da un filtro di pacchetti per contrastare questo attacco.
- 11.4 È abbastanza evidente la necessità della regola che evita la lettura verso l'alto nei sistemi di sicurezza multilivello. Motivate l'importanza del principio che evita la scrittura verso il basso.
- 11.5 Nella Figura 11.5 il collegamento instaurato dal cavallo di Troia che permette di copiare i dati ed esaminarli successivamente viene interrotto. Alice ha altre due possibilità di attacco: connettersi e tentare di leggere la stringa direttamente, oppure assegnare il livello di sicurezza sensibile al suo file. Il reference monitor è in grado di prevenire queste tipologie di attacco?

Appendice A

Aspetti della teoria dei numeri

In questo appendice forniremo alcune informazioni su due concetti cui si fa riferimento in questo libro: i numeri primi e l'aritmetica modulare.

A.1 Numeri primi e relativamente primi

In questo paragrafo, salvo notazione contraria, tratteremo solo gli interi non negativi. L'utilizzo di interi negativi non introdurrebbe differenze fondamentali.

Divisori

Si dice che $b \neq 0$ divide a se $a = mb$ per un qualche m , dove a , b , e m sono degli interi; cioè b divide a se non vi è resto nella divisione. La notazione $b|a$ è comunemente usata per indicare che b divide a . Inoltre, se $b|a$, si dice che b è *divisore* di a . Ad esempio, i divisori positivi di 24 sono 1, 2, 3, 4, 6, 8, 12 e 24.

Valgono le seguenti relazioni:

- Se $a|1$, allora $a = \pm 1$
- Se $a|b$ e $b|a$, allora $a = \pm b$
- Qualsiasi $b \neq 0$ divide 0
- Se $b|g$ e $b|h$, allora $b|(mg + nh)$ per m e n interi qualsiasi

Per verificare questo ultimo punto si noti che:

se $b|g$, allora g è della forma $g = b \times g_1$ per un qualche intero g_1

se $b|h$, allora h è della forma $h = b \times h_1$ per un qualche intero h_1

quindi

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

e perciò b divide $mg + nh$.

Numeri primi

Un intero $p > 0$ è un numero primo se i suoi soli divisori sono ± 1 e $\pm p$. I numeri primi giocano un ruolo critico nella teoria dei numeri e nelle tecniche discusse al Capitolo 3.

Un qualsiasi intero $a > 0$ può essere fattorizzato in maniera unica come:

$$a = p_1^{a_1} p_2^{a_2} \cdots p_t^{a_t}$$

dove $p_1 < p_2 < \dots < p_t$ sono numeri primi e dove ciascun a_i è un intero positivo. Ad esempio, $91 = 7 \times 13$ e $11011 = 7 \times 11^2 \times 13$.

È utile effettuare la somma in un altro modo. Se P è l'insieme di tutti i numeri primi, allora qualsiasi intero positivo può essere scritto in modo univoco nella seguente forma:

$$a = \prod_{p \in P} p^{a_p} \quad \text{dove ciascun } a_p \geq 0$$

Sul lato destro è rappresentato il prodotto su tutti i possibili numeri primi p ; per un qualche valore particolare di a , molti degli esponenti a_p saranno nulli.

Il valore di un certo intero positivo può essere specificato semplicemente elencando tutti gli esponenti non nulli della formula precedente. Quindi, l'intero 12 è rappresentato da $\{a_2 = 2, a_3 = 1\}$ e l'intero 18 da $\{a_2 = 1, a_3 = 2\}$. La moltiplicazione di due numeri è equivalente alla somma dei corrispondenti esponenti:

$$k = mn \rightarrow k_p = m_p + n_p \quad \text{per ogni } p$$

Che cosa significa, in termini di questi fattori primi, dire che $a|b$? Qualsiasi intero nella forma p^k può essere diviso solo da un intero che è di potenza minore o uguale dello stesso numero primo, p^j con $j \leq k$. Quindi

$$\underbrace{a|b}_{\omega} \rightarrow a_p \leq b_p \quad \text{per ogni } p$$

Numeri relativamente primi

Si userà la notazione $\text{MCD}(a, b)$ per indicare il massimo comune divisore di a e b . L'intero positivo c è detto massimo comune divisore di a e b se:

1. c è divisore di a e b .
2. ogni divisore di a e b è un divisore di c .

Una definizione equivalente è la seguente:

$$\text{MCD}(a, b) = \max [k, \text{ tale che } k|a \text{ e } k|b]$$

Dato che si richiede che il massimo comune divisore sia positivo, $\text{MCD}(a, b) = \text{MCD}(a, -b) = \text{MCD}(-a, b) = \text{MCD}(-a, -b)$. In generale $\text{MCD}(a, b) = \text{MCD}(|a|, |b|)$. Ad esempio, $\text{MDC}(60, 24) = 12$. Inoltre, poiché tutti gli interi non nulli dividono 0, si ha $\text{MDC}(a, 0) = |a|$.

È semplice determinare il massimo comune divisore di due interi positivi, se si esprime ciascun intero come il prodotto di numeri primi. Ad esempio:

$$300 = 2^2 \times 3^1 \times 5^2$$

$$18 = 2^1 \times 3^2$$

$$\text{MCD}(300, 18) = 2^1 \times 3^1 \times 5^0 = 6$$

In generale:

$$k = \text{MDC}(a, b) \rightarrow k_p = \min(a_p, b_p) \quad \text{per tutti i } p$$

Determinare i fattori primi di un numero grande non è un compito facile, quindi la relazione precedente non porta direttamente a un modo per calcolare il massimo comune divisore:

Gli interi a e b sono relativamente primi se non hanno fattori primi in comune, cioè se il loro solo fattore comune è 1. Questo è equivalente a dire che a e b sono relativamente primi se $\text{MDC}(a, b) = 1$. Ad esempio, 8 e 15 sono relativamente primi, in quanto i divisori di 8 sono 1, 2, 4 e 8 e i divisori di 15 sono 1, 3, 5 e 15 e quindi 1 è il solo numero in entrambi gli elenchi.

A.2 Aritmetica Modulare

Dato un qualsiasi intero positivo n e un qualsiasi intero positivo a , se si divide a con n si ottiene un quoziente intero q e un resto intero r che rispettano la seguente relazione:

$$a = qn + r \quad 0 \leq r \leq n; q = \lfloor a/n \rfloor$$

dove $\lfloor x \rfloor$ è l'intero più grande minore o uguale a x .

Nella Figura A.1 si dimostra che, dato a e un positivo n , è sempre possibile trovare q e r che soddisfano la precedente relazione. Si rappresentino gli interi sulla retta dei numeri; a cadrà in qualche punto di quella retta (viene mostrato per a positivo, ma una dimostrazione analoga può essere fatta per a negativo). Partendo da 0, si procede verso $n, 2n$ fino a qn , tale che $qn \leq a$ e $(q+1)n > a$. La distanza di qn da a è r : sono così stati trovati i due valori univoci q e r . Il resto r è spesso chiamato **residuo**.

Se a è un intero e n un intero positivo, si definisce $a \bmod n$ il resto della divisione di a con n . Quindi, per ogni intero a , si può sempre scrivere

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Due interi a e b si dicono **congruenti di modulo n** , se $(a \bmod n) = (b \bmod n)$. Questo viene scritto $a \equiv b \pmod{n}$. Ad esempio, $73 \equiv 4 \pmod{23}$ e $21 \equiv -9 \pmod{10}$. Si noti che, se $a \equiv 0 \pmod{n}$, allora $n|a$.

L'operatore modulo ha le seguenti proprietà:

1. $a \equiv b \pmod{n}$ se $n|(a - b)$
2. $(a \bmod n) = (b \bmod n)$ implica che $a \equiv b \pmod{n}$
3. $a \equiv b \pmod{n}$ implica $b \equiv a \pmod{n}$
4. $a \equiv b \pmod{n}$ e $b \equiv c \pmod{n}$ implicano $a \equiv c \pmod{n}$



Figura A.1 La relazione $a = qn + r$; $0 \leq r < n$.

Per dimostrare il primo punto, se $n|(a - b)$ allora $(a - b) = kn$ per un qualche k . Quindi si può scrivere $a = b + kn$. Ne consegue che $(a \bmod n) = (\text{resto quando } b + kn \text{ è diviso da } n) = (\text{resto quando } b \text{ è diviso da } n) = (b \bmod n)$. I punti rimanenti si possono dimostrare con la stessa facilità.

L'operatore $(\bmod n)$ fa corrispondere tutti gli interi a un insieme di interi $\{0, 1, \dots, (n-1)\}$. Perciò, alla domanda se si possano eseguire operazioni aritmetiche all'interno dei confini di questo insieme è possibile dare una risposta positiva, e questa tecnica è nota come **aritmetica modulare**.

L'aritmetica modulare presenta le seguenti proprietà:

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Si dimostri la prima proprietà. Si definisca $(a \bmod n) = r_a$ e $(b \bmod n) = r_b$. Allora si può scrivere $a = r_a + jn$, per un qualche intero j , e $b = r_b + kn$, per un qualche intero k . Allora:

$$\begin{aligned} (a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n \end{aligned}$$

Le rimanenti proprietà si dimostrano con la stessa facilità.

Appendice B

Progetti per l'insegnamento della sicurezza di rete

Estremamente diffusa tra i docenti è la pratica di assegnare agli studenti progetti di ricerca e implementazione. Questi si rivelano, infatti, come un indispensabile supporto didattico per raggiungere una chiara comprensione della sicurezza di rete. Senza di questi, potrebbe risultare difficile apprendere alcuni concetti base e le interazioni tra i componenti.

Certamente, nel testo si è cercato di presentare gli aspetti della sicurezza di rete nel modo più chiaro possibile, supportandoli anche con numerosi esercizi. I progetti saranno, pertanto, utili per consolidare l'apprendimento dei concetti introdotti nel libro, consentendo agli studenti di comprendere meglio come funzionano gli algoritmi crittografici o i protocolli, e rafforzando, nel contempo, la loro fiducia nell'essere in grado non solo di capire, ma anche di implementare una proprietà della sicurezza.

È in quest'ottica che è stato pensato e realizzato il supplemento dedicato ai docenti che adottano il testo e che potranno richiedere la password collegandosi al sito www.prenhall.com e selezionando la pagina dedicata a questo testo: Network Security Essentials, terza edizione.

In questa appendice sono fornite alcune linee guida e una descrizione sintetica del materiale di supporto cui si fa riferimento.

Il materiale di supporto include cinque tipi di progetti:

- progetti di ricerca
- progetti di programmazione
- esercizi di laboratorio
- compiti scritti
- compiti di lettura/relazione

studiati in modo da poter essere assegnati sia a gruppi di lavoro sia a singole persone.

B.1 Progetti di ricerca

Sono finalizzati a spronare gli studenti a documentarsi in letteratura o su Internet in merito ai prodotti dei fornitori, all'attività nei laboratori di ricerca e sugli sforzi nella standardizzazione.

Il supplemento per il docente comprende un suggerimento per il formato delle proposte di progetto e della relazione finale, come pure una lista di quindici possibili argomenti di ricerca. Gli studenti potranno selezionare uno degli argomenti elencati nel manuale o pure elaborarne uno originale mutuandolo dalla lista.

B.2 Progetti di programmazione

Indipendenti dalla piattaforma e dal linguaggio, i progetti di programmazione sono stati sviluppati in modo da poter essere realizzati dagli studenti utilizzando un qualsiasi computer, in un qualunque linguaggio di programmazione. Non occorre, pertanto, scaricare, installare e configurare nessuna particolare infrastruttura in quanto i progetti funzionano in modo autonomo.

Inoltre, c'è una certa flessibilità nelle dimensioni dei progetti. Quelli più impegnativi forniscono agli studenti una maggior coscienza dei risultati; quelli meno difficili possono essere assegnati in numero maggiore, offrendo l'opportunità di considerare una molteplicità di aree diverse.

Il manuale del docente include un insieme di dodici possibili progetti di programmazione.

Tra i docenti che hanno assegnato progetti di ricerca e programmazione prendendo spunto dal manuale del docente ci sono Henning Schulzrinne della Columbia University; Cetin Kaya Koc della Oregon State University e David M. Balenson del Trusted Information Systems and George Washington University.

B.3 Esercizi di laboratorio

Preparati dai professori Sanjay Rao e Ruben Torres della Purdue University, questi progetti di implementazione sono stati ideati per essere programmati su Linux, ma potrebbero essere adattati per qualunque ambiente UNIX. Loro finalità è quella di fornire un'esperienza realistica nell'implementazione delle funzioni e delle applicazioni di sicurezza.

B.4 Compiti scritti

Il loro obiettivo è quello di esercitare un effetto moltiplicatore sul processo di apprendimento di discipline tecniche come la crittografia e la sicurezza di rete, consentendo agli studenti di formarsi un'opinione più dettagliata e completa su un particolare argomento, superando così il rischio di un approccio semplicistico alla materia, limitato all'impiego di tecniche di risoluzione dei problemi, senza una reale, approfondita comprensione della materia.

Il supplemento per il docente contiene svariate proposte per compiti scritti, organizzati per capitolo.

B.5 Compiti di lettura/relazione

Infine, il materiale di supporto didattico per il docente comprende la proposta di un elenco di articoli, uno o due per capitolo, da assegnare agli studenti per rafforzare i concetti presentati durante il corso. Tutti gli articoli contenuti nel manuale sono disponibili via Internet.

GLOSSARIO

Le definizioni segnalate da un asterisco sono tratte dall'Internet Security Glossary [RFC 2828].

algoritmo RSA Algoritmo di cifratura a chiave pubblica basato sull'utilizzo della funzione potenza e dell'aritmetica modulare. È il solo algoritmo generalmente diffuso nella pratica per la cifratura a chiave pubblica.

aritmetica modulare Tipo di aritmetica intera che riconduce tutti i numeri a uno solo, che fa parte di un insieme fissato $[0 \dots n-1]$ per un qualche numero n . Qualsiasi intero al di fuori di questo intervallo si può ridurre a un intero dell'intervallo prendendo il resto della sua divisione con n .

attacco alla sicurezza Assalto al sistema di sicurezza che deriva da un atto intelligente, cioè un tentativo deliberato (specialmente nel senso di un metodo o di una tecnica) per eludere i servizi di sicurezza e violare le politiche di sicurezza di un sistema.

attacco birthday Tramite questo attacco critto-analitico si cerca di trovare due valori nel dominio di una funzione che corrispondono a uno stesso valore nel codominio della funzione stessa.

attacco di replay Attacco in cui un servizio, precedentemente autorizzato e completato, è falsificato mediante un'ulteriore "richiesta duplicata" nel tentativo di rieseguire comandi autorizzati.

attacco man-in-the-middle Forma di attacco attivo di intercettazione nella quale l'attaccante capta e modifica in modo selettivo i dati scambiati, al fine di farsi passare per una o più delle entità coinvolte nella comunicazione.

attacco meet-in-the-middle Tramite questo attacco critto-analitico si cerca di trovare un numero in ciascun dominio e codominio della composizione di due funzioni, in modo tale che il valore della prima funzione, calcolata in uno dei numeri trovati, è uguale al valore assunto dall'inverso della seconda funzione, calcolato nell'altro numero. Ciò equivale a trovarsi nel mezzo della funzione composta.

autenticatore Informazione aggiuntiva accodata a un messaggio per consentire al destinatario la verifica dell'autenticità del messaggio. L'autenticatore può essere funzionalmente indipendente dal contenuto del messaggio (per esempio, valore creato ad hoc, vedi *nonce*) o può essere funzionalmente dipendente dal contenuto del messaggio (per esempio, valore calcolato mediante una funzione hash oppure una somma di controllo crittografica).

autenticazione* Processo utilizzato per verificare l'integrità dei dati trasmessi, in particolare dei messaggi.

batteri Programma che auto-replicandosi consuma risorse di sistema.

bomba logica Logica incorporata in un programma che controlla il verificarsi di un certo insieme di condizioni nel sistema. Al loro verificarsi, il programma esegue funzionalità che portano ad azioni non autorizzate nel sistema.

byte Sequenza di otto bit, chiamata anche otetto.

canale nascosto Canale di comunicazione che rende possibile il trasferimento di informazioni secondo modalità non previste dai progettisti dell'infrastruttura di comunicazione.

cavallo di Troia* Programma con apparente o reale funzione di utilità che contiene ulteriori funzionalità (nasconde) che fraudolentemente sfruttano le autorizzazioni legittime del processo invocante al fine di violare la sicurezza del sistema.

centro per la distribuzione di chiavi (KDC) Sistema autorizzato per la trasmissione di chiavi di sessione temporanea tra due principal in comunicazione. Ciascuna chiave di sessione è trasmessa in forma cifrata, utilizzando una chiave principale (chiave master) condivisa tra il KDC e il principal coinvolto.

certificato a chiave pubblica Consiste in una chiave pubblica e in un identificativo utente del proprietario della chiave: questo blocco di informazioni viene poi firmato da una terza parte fidata. Tipicamente la terza parte è un'autorità certificativa (CA) che viene considerata fidata dalla comunità degli utenti, ad esempio, una pubblica amministrazione o un'istituzione finanziaria.

checksum crittografico Somma di controllo crittografica. Strumento di autenticazione che consiste in una funzione crittografica sia dei dati da autenticare sia della chiave segreta. Chiamato anche codice di autenticazione di messaggio (MAC).

chiave di sessione Chiave temporanea di cifratura utilizzata tra due parti durante una sessione di comunicazione.

chiave master Chiave principale; utilizzata per codificare le chiavi di sessione nella trasmissione da KDC a un principal. Tipicamente le chiavi master sono distribuite tramite strumenti di tipo non crittografico. Chiamata anche chiave di cifratura di chiavi (*key-encrypting key*).

chiave privata Una delle due chiavi utilizzate nei sistemi di cifratura asimmetrica. Al fine di assicurare comunicazione sicura, la chiave privata deve essere nota solo al suo creatore.

chiave pubblica Una delle due chiavi utilizzate nei sistemi di cifratura asimmetrica. La chiave pubblica è nota a tutti, e deve essere utilizzata congiuntamente a una corrispondente chiave privata.

chiave segreta Utilizzata in un sistema di cifratura simmetrica in cui entrambi i partecipanti devono condividere la stessa chiave; questa deve rimanere segreta per proteggere la comunicazione.

cifrario Algoritmo per cifratura e decifratura. Un cfrario sostituisce un frammento di informazione (un elemento nel testo in chiaro) con un altro oggetto, allo scopo di nasconderne il significato. Tipicamente, la regola di trasformazione utilizza una chiave segreta.

cifrario a blocchi Algoritmo simmetrico di cifratura in cui un blocco di testo in chiaro di dimensioni elevate (tipicamente 64 bit) è interamente trasformato in un blocco di testo cifrato della stessa lunghezza.

cifratura Conversione di un testo in chiaro o di un dato in un formato incomprensibile mediante un processo di traduzione reversibile basato su una tabella di conversione o un algoritmo.

cifratura a chiave pubblica Cfr *cifratura asimmetrica*.

cifratura a flusso Algoritmo di cifratura simmetrica in cui il testo cifrato risultante è prodotto bit a bit o byte a byte, considerando un flusso di caratteri del testo in chiaro senza una predeterminata lunghezza.

cifratura asimmetrica Sistema crittografico in cui il processo di cifratura e quello di decifratura sono effettuate utilizzando due chiavi diverse, una denominata chiave pubblica e l'altra chiave privata. Chiamata anche cifratura a chiave pubblica.

cifratura convenzionale Cfr *cifratura simmetrica*.

cifratura multipla Uso ripetuto di una funzione di cifratura con chiavi differenti al fine di generare corrispondenze più complesse fra testo in chiaro e testo cifrato.

cifratura simmetrica Sistema crittografico in cui il processo di cifratura e quello di decifratura sono realizzati utilizzando la stessa chiave. Detta anche cifratura convenzionale.

classe di resti Tutti gli interi che hanno lo stesso resto, quando vengono divisi per n , formano una classe di resti ($\text{mod } n$). Quindi, per un dato resto r , la classe di resti ($\text{mod } n$) cui appartiene, consiste negli interi $r, r \pm n, r \pm 2n, \dots$

codice Regola fissata (invariabile) per la sostituzione di un frammento di informazione (per esempio, lettera, parola, frase) con un altro oggetto non necessariamente della stessa natura. Generalmente, lo scopo di un codice non è quello di nascondere il significato. Esempi di codici includono il codice ASCII per la rappresentazione dei caratteri (ogni carattere è rappresentato mediante 7 bit) e il codice frequency-shift keying (ogni valore binario è rappresentato mediante una particolare frequenza).

codice di autenticazione di messaggio (MAC) Cfr *checksum crittografico*.

computazionalmente sicuro Sicuro perché il tempo e/o il costo richiesto per la vanificazione della sicurezza è troppo elevato per essere proponibile.

concatenazione tra blocchi Tecnica utilizzata durante la cifratura simmetrica a blocchi che produce un blocco in uscita dipendente non solo dal blocco di testo in chiaro corrente in ingresso e dalla chiave ma anche dagli ingressi e/o dalle uscite precedenti. L'effetto della concatenazione tra blocchi è che due istanze dello stesso blocco di testo in chiaro in ingresso generano differenti blocchi cifrati, rendendo la crittoanalisi più difficile.

confusion Scompiglio. Tecnica crittografica che cerca di rendere il più possibile complessa la relazione fra statistiche del testo cifrato e valore della chiave di cifratura. Questo è ottenuto mediante l'utilizzo di un algoritmo complesso di scrambling che dipende dalla chiave e dal testo in ingresso.

controllo dell'accesso discrezionale* Meccanismo per limitare gli accessi a oggetti basato su identità dei soggetti e/o dei gruppi cui i soggetti appartengono. Si parla di controllo discrezionale in quanto un soggetto in possesso di un certo privilegio di accesso è autorizzato a trasferire tale privilegio (anche indirettamente) a qualunque altro soggetto (a meno di restrizioni imposte dal controllo dell'accesso mandatorio).

controllo dell'accesso mandatorio Meccanismo per limitare gli accessi agli oggetti basato su classificazione predefinita di soggetti e oggetti nel sistema. Si parla di controllo mandatorio in quanto le regole di accesso non possono essere modificate dai soggetti o dai loro programmi.

crittoanalisi Branca della crittologia che ha come scopo sia l'analisi di un cifrario per risalire all'informazione originaria sia la generazione di informazione cifrata contraffatta che possa essere accettata come autentica.

crittoanalisi differenziale Tecnica in cui testi in chiaro selezionati, contenenti specifici pattern di OR esclusivo, sono cifrati. I vari pattern nel testo cifrato risultante forniscono informazioni che possono essere usate allo scopo di determinare la chiave di cifratura.

crittografia Branca della crittologia che ha come scopo la progettazione di algoritmi di cifratura e decifrazione, al fine di garantire la segretezza e/o l'autenticità dei messaggi.

crittologia Scienza che ha lo scopo di studiare comunicazioni sicure. Comprende crittografia e crittoanalisi.

decifratura Trasformazione di testo/dato cifrato (denominato testo cifrato) nel corrispondente testo/dato originario (denominato testo in chiaro).

diffusion Diffusione. Tecnica crittografica che cerca di nascondere la struttura statistica del testo in chiaro facendo in modo che l'effetto di ciascuna cifra del testo in chiaro sia ripartito su molte cifre del testo cifrato.

digest di messaggio Sintesi cifrata di un messaggio. Funzione hash.

digram Gruppo di due caratteri. In inglese e in altre lingue, la frequenza relativa di digram diversi nel testo in chiaro può essere utilizzata nella crittoanalisi di alcuni cifrari. Chiamato anche digraph.

divisore Un intero si dice divisore di un altro intero se la divisione non dà resto.

effetto valanga Caratteristica di un algoritmo di cifratura secondo cui un piccolo cambiamento nel testo in chiaro o nella chiave dà origine a un significativo cambiamento nel testo cifrato. Nel caso di codice hash, l'effetto valanga è una caratteristica secondo cui un piccolo cambiamento nel messaggio origina un significativo cambiamento nel digest di messaggio.

firewall Computer dedicato che funge da interfaccia con gli elaboratori esterni a una rete e che è dotato di speciali misure di sicurezza per proteggere le informazioni sensibili sui computer interni alla rete. Viene usato sulle connessioni di rete verso l'esterno, specialmente su Internet, e sulle linee dial-up.

firma digitale Meccanismo di autenticazione che consente al creatore di un messaggio di apporvi un codice che agisce da firma. Tale firma garantisce la fonte di provenienza e l'integrità del messaggio.

funzione hash Funzione che mette in corrispondenza un blocco dati di lunghezza variabile o un messaggio con un valore di lunghezza fissa, denominato codice hash. La funzione è concepita in modo tale che, quando protetta, fornisce uno strumento di autenticazione per i dati o il messaggio. Chiamata anche digest di messaggio.

funzione unidirezionale Funzione di facile computazione, ma per la quale risulta improponibile il calcolo dell'inversa.

funzione unidirezionale con trapdoor Funzione di facile computazione, ma per la quale risulta improponibile il calcolo dell'inversa, a meno di conoscere informazioni riservate.

generatore di numeri pseudocasuali Funzione che genera in modo deterministico una sequenza di numeri che sono apparentemente statisticamente casuali.

honeypot Sistemi che servono da esca, progettati per indurre un potenziale attaccante ad abbandonare i sistemi critici. Costituiscono una forma di sistema per la rilevazione delle intrusioni.

incondizionatamente sicuro Sicuro anche rispetto ad attacchi perpetrati con illimitate risorse computazionali e di tempo.

intruso Individuo che acquisisce, o tenta di acquisire, accessi o privilegi non autorizzati in un sistema di elaborazione.

kerberos Nome del servizio di autenticazione nell'ambito del progetto Athena.

massimo comune divisore Il più grande divisore comune di due interi, a e b , è l'intero positivo più grande che divide sia a che b . Si dice che un intero divide un altro intero se la divisione non ha resto.

meccanismo di sicurezza Processo (o dispositivo che incorpora tale processo) progettato per rilevare, prevenire o ripristinare il sistema da un attacco alla sicurezza.

minaccia alla sicurezza Potenziale violazione di sicurezza, che diventa realtà quando si verificano le circostanze, le capacità, le azioni o gli eventi che possono aprire una breccia nella sicurezza e infliggere danni. Vale a dire che una minaccia è un potenziale pericolo che potrebbe sfruttare una vulnerabilità.

modalità operativa Tecnica per aumentare l'effetto di un algoritmo crittografico o per adattare l'algoritmo a un'applicazione come, ad esempio, quella relativa a un cifrario a blocchi, a una sequenza di blocchi di dati o a un flusso di dati.

nibble Una sequenza di quattro bit.

nonce Identificatore o numero utilizzato una sola volta.

password* Stringa di caratteri utilizzata per autenticare l'identità di un utente. La conoscenza della password e dell'identificatore utente ad essa associato è considerata evidenza di autorizzazione all'utilizzo delle risorse del sistema secondo le autorizzazioni specificate per l'identificatore utente considerato.

radice primitiva Se r e n sono interi relativamente primi con $n > 0$, e se $\phi(n)$ è il più piccolo esponente positivo m tale che $r^m \equiv 1 \pmod{n}$, allora r viene detta radice primitiva modulo n .

resto Quando un intero a viene diviso da un intero n , ciò che rimane r viene chiamato resto. In modo equivalente, $r = a \bmod n$.

rete privata virtuale Consiste in un insieme di computer che sono interconnessi tramite una rete relativamente insicura e che fanno uso della cifratura e di protocolli speciali per fornire la sicurezza.

servizio di sicurezza Servizio di elaborazione o di comunicazione che aumenta la sicurezza di un sistema di elaborazione dei dati e dei trasferimenti di informazioni di un'organizzazione. Il servizio ha lo scopo di contrastare gli attacchi alla sicurezza e fa uso di uno o più meccanismi di sicurezza per erogare il servizio stesso.

sicurezza multilivello Insieme degli strumenti che realizzano il controllo dell'accesso in presenza di dati con molteplici livelli di classificazione.

sistema di rilevazione delle intrusioni Insieme di strumenti automatizzati progettati per rilevare accessi non autorizzati al sistema dell'host.

sistema trusted Sistema fidato. Sistema di elaborazione/operativo in cui può essere verificata e comprovata la realizzazione di una certa politica di sicurezza.

steganografia Metodi per nascondere l'esistenza di un messaggio o di altri dati. È differente dalla crittografia che nasconde il significato di un messaggio ma non nasconde il messaggio stesso.

testo cifrato Risultato di un algoritmo di cifratura; forma cifrata di un messaggio o di un dato.

testo in chiaro Ingresso di una funzione di cifratura o uscita di una funzione di decifratura.

trapdoor Punto di ingresso segreto e non documentato di un programma, utilizzato per consentire l'accesso evitando i normali meccanismi di accesso e autenticazione.

vettore di inizializzazione Blocco casuale di dati utilizzato in fase di avvio della procedura di cifratura di molti blocchi di testo in chiaro, quando si adotta la tecnica di cifratura basata sulla concatenazione tra blocchi. Il vettore di inizializzazione ha lo scopo di sventare attacchi basati su testo in chiaro conosciuto. (cfr Tab. 2.1)

virus Codice inserito all'interno di un programma che si diffonde nel sistema inserendo sue copie in uno o più programmi. In aggiunta alla replicazione, il virus solitamente compie alcune azioni indesiderate nel sistema.

worm Programma in grado di replicarsi e inviare copie di se stesso su reti di comunicazione. Alla ricezione, il programma worm può essere attivato per replicarsi e propagarsi nuovamente. In aggiunta alla diffusione, il programma worm solitamente compie alcune azioni indesiderate nel sistema.

BIBLIOGRAFIA

- ALVA90 Alvare A., "How Crackers Crack Passwords or What Passwords to Avoid" in *Proceedings, UNIX Security Workshop II*, agosto 1990.
- ANDE80 Anderson J., *Computer Security Threat Monitoring and Surveillance*, Fort Washington (PA), James P. Anderson Co., aprile 1980.
- AUDI04 Audin G., "Next-Gen Firewalls: What to Expect" in *Business Communications Review*, giugno 2004.
- AXEL00 Axelsson S., "The Base-Rate Fallacy and the Difficulty of Intrusion Detection" in *ACM Transactions and Information and System Security*, agosto 2000.
- BACE00 Bace R., "Intrusion Detection" in *Macmillan Technical Publishing*, Indianapolis, 2000.
- BACE01 Bace R. e Mell E., "Intrusion Detection Systems" in *NIST Special Publication SP 800-31*, novembre 2000.
- BARR03 Barreto E. e Rijmen V., "The Whirlpool Hashing Function" in *Submitted to NESSIE*, settembre 2000, rev. maggio 2003.
- BAUE88 Bauer D. e Koblentz M., "NIDX – An Expert System for Real-Time Network Intrusion Detection" in *Proceedings, Computer Networking Symposium*, aprile 1988.
- BELL90 Bellovin S. e Merritt M., "Limitations of the Kerberos Authentication System" in *Computer Communications Review*, ottobre 1990.
- BELL92 Bellovin S., "There Be Dragons" in *Proceedings, UNIX Security Symposium III*, settembre 1992.
- BELL93 Bellovin S., "Packers Found on an Internet" in *Computer Communications Review*, luglio 1993.
- BELL94 Bellovin S. e Cheswick W., "Network Firewalls" in *IEEE Communications Magazine*, settembre 1994.
- BELL96a Bellare M., Canetti R. e Krawczyk H., "Keying Hash Functions for Message Authentication" in *Proceedings, CRYPTO '96*, agosto 1996, Springer-Verlag.
Disponibile all'indirizzo <http://www-cse.ucsd.edu/users/mihir>
- BELL96b Bellare M., Canetti R. e Krawczyk H., "The HMAC Construction" in *CryptoBytes*, primavera 1996.
- BERS92 Berson T., "Differential Cryptanalysis Mod 232 with Applications to MD5" in *Proceedings, EUROCRYPT 92*, maggio 1992, Springer-Verlag.
- BISH03 Bishop M., *Computer Security: Art and Science*, Boston, Addison-Wesley, 2003.
- BISH05 Bishop M., *Introduction to Computer Security*, Boston, Addison-Wesley, 2005.
- BLOO70 Bloom B., "Space/time Trade-Offs in Hash Coding with Allowable Errors" in *Communications of the ACM*, luglio 1970.
- BLUM97a Blumenthal U., Hien N. e Wijnen B., "Key Derivation for Network Management Applications" in *IEEE Network*, maggio/giugno 1997.

- BLUM97b Blumenthal U. e Wijnen B., "Security Features for SNMPv3" in *The Simple Times*, dicembre 1997.
- BOER93 Boer B. e Bosselaers A., "Collisions for the Compression Function of MD5" in *Proceedings, EUROCRYPT '93*, 1993, Springer-Verlag.
- BRYA88 Bryant W., *Designing an Authentication System: A Dialogue in Four Scenes*, documento del Progetto Athena, febbraio 1988.
Disponibile all'indirizzo <http://web.mit.edu/kerberos/www/dialogue.html>
- CASS01 Cass S., "Anatomy of Malice" in *IEEE Spectrum*, novembre 2001.
- CERT01 CERT Coordination Center, "Denial of Service Attacks", giugno 2001.
Disponibile all'indirizzo http://www.cert.org/tech_tips/denial_of_service.html
- CERT02 CERT Coordination Center, "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol" in *CERT Advisory CA-2002-03*, 25 giugno 2002.
Disponibile all'indirizzo www.cert.org/advisories/CA-2002-03.html
- CHAN02 Chang R., "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial" in *IEEE Communications Magazine*, ottobre 2002.
- CHAP00 Chapman D. e Zwicky E., *Building Internet Firewalls*, Sebastopol (CA), O'Reilly, 2000.
- CHEN98 Cheng P. et al., "A Security Architecture for the Internet Protocol" in *IBM Systems journal*, n. 1, 1998.
- CHES97 Chess D., "The Future of Viruses on the Internet" in *Proceedings, Virus Bulletin International Conference*, ottobre 1997.
- CHES03 Cheswick W. e Bellovin, *S. Firewalls and Internet Security: Repelling the Wily Hacker*, Reading (MA), Addison-Wesley, 2003.
- COHE94 Cohen F., *A Short Course on Computer Viruses*, New York, Wiley, 1994.
- CORM01 Cormen T., Leiserson C., Rivest R. e Stein C., *Introduction to Algorithms*, Cambridge (MA), MIT Press, 2001.
- DAMG89 Damgård I., "A Design Principle for Hash Functions" in *Proceedings, CRYPTO '89*, 1989, Springer-Verlag.
- DAV189 Davies D. e Price W., *Security for Computer Networks*, New York, Wiley, 1989.
- DAVI93 Davies C. e Ganesan R., "BApasswd: A New Proactive Password Checker" in *Proceedings, 16th National Computer Security Conference*, settembre 1993.
- DAWS96 Dawson E. e Nielsen L., "Automated Cryptoanalysis of XOR Plaintext Strings" in *Cryptologia*, aprile 1996.
- DENN87 Denning D., "An Intrusion-Detection Model" in *IEEE Transactions on Software Engineering*, febbraio 1987.
- DJFF76 Diffie W. e Hellman M., "Multiuser Cryptography Techniques" in *IEEE Transactions on Information Theory*, novembre 1976.
- DIFF88 Diffie W., "The First Ten Years of Public-Key Cryptography" in *Proceedings of the IEEE*, maggio 1988, ristampato in [SIMM92a].
- DOBB96a Dobbertin H., "The Status of MD5 After a Recent Attack" in *CryptoBytes*, estate 1996.
- DORA03 Doraswamy N. e Harkins D., *IPSec*, Upper Saddle River (NJ), Prentice Hall, 2003.
- DREW99 Drew G., *Using SET for Secure Electronic Commerce*, Upper Saddle River (NJ), Prentice Hall, 1999.
- EFF98 Electronic Frontier Foundation, *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*, Sebastopol (CA), O'Reilly, 1998.
- ENGE80 Enger N. e Howerton P., *Computer Security*, New York, Amacom, 1980.
- FEIS73 Feistel H., "Cryptography and Computer Privacy" in *Scientific American*, maggio 1973.
- FELT03 Felten E., "Understanding Trusted Computing: Will Its Benefits Outweigh its Drawbacks?" in *IEEE Security and Privacy*, maggio/giugno 2003.

- FLUH01 Fluhrer S. e McGrew D., "Statistical Analysis of the Alleged RC4 Key Stream Generator" in *Proceedings, Fast Software Encryption 2000*, 2000.
- FLUH01 Fluhrer S., Mantin I. e Shamir A., "Weakness in the Key Scheduling Algorithm of RC4" in *Proceedings, Workshop in Selected Areas of Cryptography*, 2001.
- FORD95 Ford W., "Advances in Public-Key Certificate Standards" in *ACM SIGSAC Review*, luglio 1995.
- FORR97 Forrest S., Hofmeyr S. e Somayaji A., "Computer Immunology" in *Communications of the ACM*, ottobre 1997.
- FRAN01 Frankel S., *Demystifying the IPsec Puzzle*, Boston, Artech House, 2001.
- GARD77 Gardner M., "A New Kind of Cipher That Would Take Millions of Years to Break" in *Scientific American*, agosto 1977.
- GARF97 Garfinkel S., Spafford G., *Web Security & Commerce*, Cambridge (MA), O'Reilly and Associates, 1997.
- GASS88 Gasser M., *Building a Secure Computer System*, New York, Van Nostrand Reinhold, 1988.
- GAUD00 Gaudin S., "The Omega Files" in *Network World*, giugno 2000.
- GOLL99 Gollmann D., *Computer Security*, New York, Wiley, 1999.
- GUTM02 Gutmann E., "PKL It's Not Dead, Just Resting" in *Computer*, agosto 2002.
- HARL01 Harley D., Slade R. e Gattiker U., *Viruses Revealed*, New York, Osborne/McGraw Hill, 2001.
- HEBE92 Heberlein L., Mukherjee B. e Levitt K., "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks" in *Proceedings, 15th National Computer Security Conference*, ottobre 1992.
- HELD96 Held G., *Data and Image Compression: Tools and Techniques*, New York, Wiley, 1996.
- HONE01 The Honeynet Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*, Reading (MA), Addison-Wesley, 2001.
- HUIT98 Huitema C., *IPv6: The New Internet Protocol*, Upper Saddle River (NJ), Prentice Hall, 1998.
- IANS90 I'Anson C. e Mitchell C., "Security Defects in CCITT Recommendation X.509 – The Directory Authentication Framework" in *Computer Communications Review*, aprile 1990.
- ILGU93 Ilgun K., "USTAT: A Real-Time Intrusion Detection System for UNIX" in *Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, maggio 1993.
- JAVI91 Javitz H. e Valdes A., "The SRI IDES Statistical Anomaly Detector" in *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, maggio 1991.
- JIAN02 Jiang G., "Multiple Vulnerabilities in SNMP" in *Security and Privacy Supplement to Computer Magazine*, 2002.
- JUEN85 Jueman R., Matyas S. e Meyer C., "Message Authentication" in *IEEE Communications Magazine*, settembre 1985.
- KENT00 Kent S., "On the Trail of Intrusions into Information Systems" in *IEEE Spectrum*, dicembre 2000.
- KEPH97a Kephart J., Sorkin G., Chess D. e White S., "Fighting Computer Viruses" in *Scientific American*, novembre 1997.
- KEPH97b Kephart J., Sorkin G., Swimmer B. e White S., "Blueprint for a Computer Immune System" in *Proceedings, Virus Bulletin International Conference*, ottobre 1997.
- KLEI90 Klein D., "Foiling the Cracker: A Survey of, and Improvements to, Password Security" in *Proceedings, UNIX Security Workshop II*, agosto 1990.
- KNUD98 Knudsen L. et al., "Analysis Method for Alleged RC4" in *Proceedings, ASIACRYPT '98*, 1998.
- KOBL92 Koblas D. e Koblas M., "SOCKS" in *Proceedings, UNIX Security Symposium III*, settembre 1992.

- KOHL89 Kohl J., "The Use of Encryption in Kerberos for Network Authentication" in *Proceedings, CRYPTO '89*, 1989, Springer-Verlag.
- KOHL94 Kohl J., Neuman B. e Ts'o T., "The Evolution of the Kerberos Authentication Service" in Brazier F. e Johansen D., *Distributed Open Systems*, Los Alamitos (CA), IEEE Computer Society Press, 1994.
Disponibile all'indirizzo <http://web.mit.edu/kerberos/www/papers.html>
- KUMA97 Kumar I., *Cryptology*, Laguna Hills (CA), Aegean Park Press, 1997.
- LEUT94 Leutwyler K., "Superhack" in *Scientific American*, luglio 1994.
- LODI98 Lodin S., Schuba C., "Firewalls Fend Off Invasions from the Net" in *IEEE Spectrum*, febbraio 1998.
- LUNT88 Lunt T. e Jagannathan R., "A Prototype Real-Time Intrusion-Detection Expert System" in *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, aprile 1988.
- MACG97 Macgregor R., Ezvan C., Liguori L. e Han J., *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*, IBM RedBook SG24-4978-00, 1997.
Disponibile all'indirizzo www.redbooks.ibm.com
- MADS93 Madsen J., "World Record in Password Checking" in *Usenet, comp.security.miscnewsgroup*, 18 agosto 1993.
- MANT01 Mantin I. e Shamir A., "A Practical Attack on Broadcast RC4" in *Proceedings, Fast Software Encryption*, 2001.
- MARK97 Markham T., "Internet Security Protocol" in *Dr. Dobb's Journal*, giugno 1997.
- MCHU00 McHugh J., Christie A. e Allen J., "The Role of Intrusion Detection Systems" in *IEEE Software*, settembre/ottobre 2000.
- MEIN01 Meinel C., "Code Red for the Web" in *Scientific American*, ottobre 2001.
- MENE97 Menezes A., Oorschot P. e Vanstone S., *Handbook of Applied Cryptography*, Boca Raton (FL), CRC Press, 1997.
- MERK79 Merkle R., *Secrecy, Authentication, and Public Key Systems*, tesi di dottorato, Stanford University, giugno 1979.
- MERK89 Merkle R., "One Way Hash Functions and DES" in *Proceedings, CRYPTO '89*, 1989, Springer-Verlag.
- MEYE82 Meyer C. e Matyas S., *Cryptography: A New Dimension in Computer Data Security*, New York, Wiley, 1982.
- MILL88 Miller S., Neuman B., Schiller J. e Saltzer J., "Kerberos Authentication and Authorization System" in *Section E.2.1, Technical Plan del Progetto Athena*, MIT Progetto Athena, Cambridge (MA), 27 ottobre 1988.
- MIRK04 Mirkovic J. e Relher, P., "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms" in *ACM SIGCOMM Computer Communications Review*, aprile 2004.
- MIST98 Mister S. e Tavares S., "Cryptanalysis of RC4-Like Ciphers" in *Proceedings, Workshop in Selected Areas of Cryptography*, SAC '98, 1998.
- MITC90 Mitchell C., Walker M. e Rush D., "CCITT/ISO Standards for Secure Message Handling" in *IEEE Journal on Selected Areas in Communications*, maggio 1989.
- MOOR01 Moore M., "Inferring Internet Denial-of-Service Activity" in *Proceedings of the 10th USENIX Security Symposium*, 2001.
- NACH97 Nachenberg C., "Computer Virus-Antivirus Coevolution" in *Communications of the ACM*, gennaio 1997.
- NEED78 Needham R. e Schroeder M., "Using Encryption for Authentication in Large Networks of Computers" in *Communications of the ACM*, dicembre 1978.

- NING04 Ning E. et al., "Techniques and Tools for Analyzing Intrusion Alerts" in *ACM Transactions on Information and System Security*, maggio 2004.
- OPPL97 Oppiger R., "Internet Security: Firewalls and Beyond" in *Communications of the ACM*, maggio 1997.
- OPPL05 Oppiger R. e Rytz R., "Does Trusted Computing Remedy Computer Security Problems?" in *IEEE Security and Privacy*, marzo/aprile 2005.
- PATR04 Patrikakis C., Masikos M. e Zouraraki O., "Distributed Denial of Service Attacks" in *The Internet Protocol Journal*, dicembre 2004.
- PAUL03 Paul S. e Preneel B., "Analysis of Non-fortuitous Predictive States of the RC4 Keystream Generator" in *Proceedings, INDOCRYPT '03*, 2003.
- PAUL04 Paul S. e Preneel B., "A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher" in *Proceedings, Fast Software Encryption*, 2004.
- PERL99 Perlman R., "An Overview of PKI Trust Models" in *IEEE Network*, novembre/dicembre 1999.
- PFLE03 Pfleeger C., *Security in Computing*, Upper Saddle River (NJ), Prentice Hall, 2003.
- PIAT91 Piattelli-Palmarini M., "Probability: Neither Rational nor Capricious" in *Bostonia*, marzo 1991.
- PIEP03 Pieprzyk J., Hardjono T. e Seberry J., *Fundamentals of Computer Security*, New York, Springer-Verlag, 2003.
- PORR92 Porras P., *STAT: A State Transition Analysis Tool for Intrusion Detection*, tesi di laurea, University of California at Santa Barbara, luglio 1992.
- PREN02 Preneel B., "New European Schemes for Signature, Integrity and Encryption (NESSIE): A Status Report" in *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems: Public Key Cryptography*, 2002.
- PROC01 Proctor P., *The Practical Intrusion Detection Handbook*, Upper Saddle River (NJ), Prentice Hall, 2001.
- PUDO02 Pudovkina M., "Statistical Weaknesses in the Alleged RC4 Keystream Generator" in *Proceedings, 4th International Workshop on Computer Science and Information Technologies*, 2002.
- RESC01 Rescorla E., *SSL and TLS: Designing and Building Secure Systems*, Reading (MA), Addison-Wesley, 2001.
- RIVE78 Rivest R., Shamir A. e Adleman L., "A Method for Obtaining Digital Signatures and Public Key Cryptosystems" in *Communications of the ACM*, febbraio 1978.
- ROBS95a Robshaw M., *Stream Ciphers*, RSA Laboratories Technical Report TR-701, luglio 1995.
<http://www.rsasecurity.com/rsalabs>
- ROBS95b Robshaw M., *Block Ciphers*, RSA Laboratories Technical Report TR-601, agosto 1995.
<http://www.rsasecurity.com/rsalabs>
- RODR02 Rodriguez A. et al., *TCP/IP Tutorial and Technical Overview*, Upper Saddle River (NJ), Prentice Hall, 2002.
- SAFF93 Safford D., Schales D. e Less D., "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment" in *Proceedings, UNIX Security Symposium IV*, ottobre 1993.
- SCHN00 Schneier B., *Secrets and Lies: Digital Security in a Networked World*, New York, Wiley, 2000.
- SCHN96 Schneier B., *Applied Cryptography*, New York, Wiley, 1996.
- SIMM92a Siminons G. (a cura di), *Contemporary Cryptology: The Science of Information Integrity*, Piscataway (NJ), IEEE Press, 1992.
- SING99 Singh S., *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, New York, Anchor Books, 1999.

- SMIT97 Smith R., *Internet Cryptography*, Reading (MA), Addison-Wesley, 1997.
- SNAP91 Snapp S. et al., "A System for Distributed Intrusion Detection" in *Proceedings, COMPCON Spring '91*, 1991.
- SPAF92a Spafford E., "Observing Reusable Password Choices" in *Proceedings, UNIX Security Symposium III*, settembre 1992.
- SPAF92b Spafford E., "OPUS: Preventing Weak Password Choices" in *Computers and Security*, n. 3, 1992.
- STAL99 Stallings W., *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Reading (MA), Addison-Wesley, 1999.
- STAL04 Stallings W., *Computer Networking with Internet Protocols and Technology*, Upper Saddle River (NJ), Prentice Hall, 2004.
- STAL06a Stallings W., *Cryptography and Network Security: Principles and Practice*, quarta edizione, Upper Saddle River (NJ), Prentice Hall, 2006.
- STAL06b Stallings W., "The Whirlpool Secure Hash Function" in *Cryptologia*, gennaio 2006.
- STEI88 Steiner J., Neuman C. e Schiller J., "Kerberos: An Authentication Service for Open Networked Systems" in *Proceedings of the Winter 1988 USENIX Conference*, febbraio 1988.
- STEP93 Stephenson P., "Preventive Medicine" in *LAN Magazine*, novembre 1993.
- STER92 Sterling B., *The Hacker Crackdown: Law and Disorder on the Electronic Frontier*, New York, Bantam, 1992.
- STIN06 Stinson D., *Cryptography: Theory and Practice*, Boca Raton (FL), CRC Press, 2006.
- STOL88 Stoll C., "Stalking the Wily Hacker" in *Communications of the ACM*, maggio 1988, ristampato in [DENN90].
- STOL89 Stoll C., *The Cuckoo's Egg*, New York, Doubleday, 1989.
- SZOR05 Szor P., *The Art of Computer Virus Research and Defense*, Reading (MA), Addison-Wesley, 2005.
- THOM84 Thompson K., "Reflections on Trusting Trust (Deliberate Software Bugs)" in *Communications of the ACM*, agosto 1984.
- TIME90 Time, Inc. Computer Security, *Understanding Computers Series*, Alexandria (VA), Time-Life Books, 1990.
- TSUD92 Tsudik G., "Message Authentication with One-Way Hash Functions" in *Proceedings, INFOCOM '92*, maggio 1992.
- TUNG99 Tung B., *Kerberos: A Network Authentication System*, Reading (MA), Addison-Wesley, 1999.
- VACC89 Vaccaro H. e Liepins G., "Detection of Anomalous Computer Session Activity" in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, maggio 1989.
- VJA02 Vijayan J., "Denial-of-Service Attacks Still a Threat" in *ComputerWorld*, aprile 2002.
- WACK02 Wack J., Cutler K. e Pole J., *Guidelines on Firewalls and Firewall Policy*, NIST Special Publication SP 800-41, gennaio 2002.
- WAGN00 Wagner D. e Goldberg I., "Proofs of Security for the UNIX Password Hashing Algorithm" in *Proceedings, ASIACRYPT '00*, 2000.
- WANG05 Wang X., Yin Y. e Yu H., "Finding Collisions in the Full SHA-1" in *Proceedings, CRYPTO '05*, 2005, Springer-Verlag.
- WILS05 Wilson J., "The Future of the Firewall" in *Business Communications Review*, maggio 2005.
- YUVA79 Yuval G., "How to Swindle Rabin" in *Cryptologia*, luglio 1979.
- ZIV77 Ziv J. e Lempel A., "A Universal Algorithm for Sequential Data Compression" in *IEEE Transactions on Information Theory*, maggio 1977.

INDICE ANALITICO

A

- accessi
 contesto MIB, 303-304
 controllo, 12, 14, 16
 controllo, gruppo, 302
 diritto, 385
 matrice di, 385
 minacce, 18
 modalità di, 280, 308
 modello di controllo,
 basata su viste (VACM), 302
 livello di sicurezza, 303
 politica di, 278, 280
 SNMPv1, 281
 tipo di, 18, 164, 304
 vista MIB, 280, 303
- advanced encryption standard (AES), 39-43
 aggiunta di una chiave di iterazione, 40-43
 algoritmo, 40
 array Stato, 41
 cifratura, 40
 decifratura, 40
 mescolamento delle colonne, 41
 scorrimento di righe, 41
 sostituzione di byte, 41
 struttura, 40
- aggiornamento della coppia di chiavi, 130
 aggiunta di una chiave di iterazione, AES, 40-43
 algoritmi
 advanced encryption standard (AES), 40
 autenticazione, 190
 informazioni relative ad AH, 192
- buffer sliding-history, 179
 cifratura simmetrica a blocchi, 35-43
 crittografia delle curve ellittiche (ECC), 87
 data encryption (DEA), 35
 descrizione, 36
 data encryption standard (DES), 35
 di decompressione, 180
 ESP, 201-202
 firma digitale (DSA), 87
 LZ77, 178-180
 requisito MUST, 167
 MUST, S/MIME, 173
 SHOULD, 167
 SHOULD, S/MIME, 173
 scambio delle chiavi di Diffie-Hellman, 83
 triplo DES (3DES), 38
- algoritmo
 a chiave pubblica RSA, 80
 HMAC, 75
 MD5 per la generazione di digest di messaggio, 73-77
 di compressione, buffer look-ahead, 179
 di decifratura,
 definizione, 29
 principi di crittografia a chiave pubblica, 180
 di decompressione, 180
 di firma digitale (DSA, digital signature algorithm), 87
 Diffie-Hellman
 per lo scambio delle chiavi, 83
 protocolli per lo scambio delle chiavi, 85
- RC4, 45-47
 generazione del flusso, 46
 inizializzazione di S, 46

logica di, 48
robustezza di, 47
analisi del traffico, 7-8, 205, 293
anti-replay, IPSec
 finestra, 192
 servizio, 196
applicazione di generazione dei comandi, SNMPv3, 289
 risposta ai comandi, SNMPv3, 289
applicazioni di autenticazione
 lettura consigliate e siti web, 131
architettura SSL
 certificato dell'entità prioritaria, 235
 chiave di scrittura, 236
 identificatore di sessione, 235
 MAC segreto del server e del client, 236
 metodo di compressione, 236
 specificità del cifrario, 236
 valore casuale di server e client, 236
 valore segreto principale, 236
 vettori di inizializzazione, 236
architetture
 aritmetica modulare, 69, 82, 399, 400
 gestione di rete, 275
 open systems interconnection (OSI), 6, 235, 273, 377
 protocollo di gestione di rete, 271
 secure socket layer (SSL), 234
 sicurezza IP, 185
array di stato, AES, 39-43
associazione di sicurezza (SA)
 bundle, 206
 combinazioni di base, 207
 concezione di, 188
 ESP con opzione di autenticazione, 209
 parameter, 191
 selettori, 192
 transport adjacency, 206
attacchi
 alle risorse interne, 364
 che consumano le risorse per la trasmissione dei dati, 366
 con piccoli frammenti, 378
 contraffazione dell'indirizzo IP (*IP spoofing*), 378
 DDoS diretto, 366
 intradamento di sorgente, 378

man-in-the-middle, 86, 210, 267
negazione del servizio (DoS), 8, 364
negazione del servizio distribuito (DDoS), 364-369
riflettori DDoS, 366, 367
si veda anche software dannosi, attacchi alla sicurezza, minacce
attacchi alla sicurezza,
 analisi del traffico, 8, 293
 attivi, 7
 definizione, 7
 masquerade, 7
 modifica dei messaggi, 7
 negazione del servizio, 8, 364
 passivi, 7
 replay, 7
attacchi delle password, Kerberos, 114
attacchi di negazione del servizio (DoS), 8, 364
 definizione, 364
 distribuiti (DDoS), 364-369
 USM, mancanza di sicurezza, 281
 si veda anche sicurezza web
attacchi di negazione del servizio distribuiti (DDoS)
 attacco che consuma le risorse per la trasmissione dei dati, 366
 costruzione di una rete di attacco, 368
 DDoS diretto, 366
 descrizione, 364
 riflettori, 366, 367
 rimedi per, 369
attacco
 DDOS con riflettori, 366, 367
 man-in-the-middle, 86, 210, 267
 scambio di chiavi di Diffie-Hellman, 86
autenticazione
 algoritmi, ESP, 209
 applicazioni di, 99
 basata su cifratura convenzionale, 61-62
 campo Data, ESP, 196, 201
 infrastruttura per la chiave pubblica (PKI), 129
 Kerberos, 99-118
 protocollo Oakley per la scelta delle chiavi, 210-213
 senza cifratura del messaggio, 62

autenticazione dei messaggi
 approcci alla, 61-66
 cifratura e, 61
 codice (MAC), 62-64
 crittografia a chiave pubblica, principi, 76
 firmi digitali, 88
 funzioni hash
 semplici, 67-69
 sicure, 66-76
 unidirezionali, 64-66
 gestione delle chiavi, 89-91
 lettura consigliate e siti web, 91
autorità certificativi (CA, *certification authority*),
 infrastruttura per la chiave pubblica (PKI), 129
autorità di registrazione (RA, *registration authority*),
 infrastruttura per la chiave pubblica (PKI), 130
azione del giorno zero, worm, 359

B

base di dati delle politiche di sicurezza (*security policy database*), IPSec, 192
base-rate fallacy
 indipendenza, 343
 probabilità condizionale, 343
 rilevazione delle intrusioni e, 326
 teorema di Bayes, 344
basic encoding rules (BER), 113, 170
bomba logica, 348, 349

C

CA, certificati, 123
campi
 delle estensioni, 128
 ESP, 196, 201
campo
 azione, 320
 completamento 202
 ESP, 202
 condizione di eccezione, 320
content-description, S/MIME, 161
content-ID, S/MIME, 161
content Transfer-Encoding, 164

content-Transfer-Encoding, S/MIME, 161
content-Type, 161, 164
content-Type, S/MIME, 161
dati di autenticazione, 197-198
dati di identificazione, 217
dati di payload, 201
del puntatore, 180
di fiducia del proprietario, 154
firma, 120
intestazione successiva, 193
KEYLEGIT, 155
legittimità della chiave, 154
 valore, 156
lunghezza del payload, 215
message-ID, 160
numero di sequenza, 191, 196
 contatore del, 191
oggetto, 320
OWNERTRUST, 155, 156
payload, 195
scambio di chiavi, 217
signerInfo, 173
SIGTRUST, 155
soggetto, 126
timestamp, 320
utilizzo risorse, 320
versione, 198

CC, *si veda criteri comuni*

certificati, 120
 a chiave pubblica, 89
 autenticazione, unidirezionale, 125
 bidirezionale, 126
 tridirezionale, 126
autorità certificativa (CA), 89, 119, 123
certificati, crittografia a chiave pubblica, 89

di tipo
 forward, 123
 reverse, 123
estensioni, 120
firma, 120
identificatore, dell'algoritmo di firma, 120
 unico, 120
informazioni sulle chiavi, 120
numero di serie, 120
periodo di validità, 120
procedure di autenticazione, 124
revoca dei, 124

uso dei certificati a chiave pubblica, 90
utente, ottenere un, 122
versione, 120
chiavi
di sessione, generazione, PGP, Kerberos, 114
pubbliche, crittografia a chiave pubblica, 78
secrete, crittografia a chiave pubblica, 89
cifrario a blocchi, definizione, 31
modalità cipher block chaining (CBC), 49
modalità cipher feedback (CFB), 50
modalità operative, 47-52
vettore di inizializzazione, 50
cifrario a flusso
algoritmo RC4, 43-45
definizione, 31
cifratura
advanced encryption standard (AES), 40
ESP, 203
posizionamento dei dispositivi di, 52-54
struttura del cifrario di Feistel, 33-35
cifratura a chiave pubblica
gestione delle chiavi, 213
cifratura simmetrica
a blocchi, algoritmi, 35-43
algoritmo di cifratura, 29
algoritmo di decifratura, 29
chiave segreta, 29
testo cifrato, 29
testo in chiaro, 29
struttura del cifrario di Feistel, 33-35
cifratura simmetrica e riservatezza dei messaggi,
lettura consigliate, 57
siti web, 57
classe, criteri comuni (CC), 390
clear signing, S/MIME, 172-173
codice di autenticazione del messaggio (MAC),
62-64
tecnica, 65
TLS, 74
compiti
di lettura/relazione, sicurezza di rete, 402
scritti, sicurezza di rete, 402
completamento, campo, 202
componente, criteri comuni (CC), 390
contatore
del campo numero di sequenza, 191

di sequenza, overflow del, 191
rilevazione delle intrusioni, 321
controllo dell'accesso, modello di controllo dell'accesso basato su viste (VACM), 302
controlli discrezionali degli accessi, criteri comuni (CC), 391
content-description, S/MIME, campo, 161
content-ID, S/MIME, campo, 161
content Transfer-Encoding, campo, 164
content-Transfer-Encoding, S/MIME, campo, 161
content-Type, campo, 161, 164
content-Type, S/MIME, campo, 161
criteri comuni (CC)
classe, 390
componente, 390
controlli discrezionali degli accessi, 391
famiglia, 390
obiettivi, 392
di sicurezza (ST, *security target*), 394
profili, 392
di protezione (PP, *protection profiles*), 392
requisiti, 390, 391
di garanzia, 390
funzionali, 390, 394
target of evaluation (TOE, oggetto della valutazione), 390
crittoanalisi, 31-33
crittografia a chiave pubblica
algoritmi, 80-88
algoritmo di cifratura, 77
algoritmo di decifratura, 77
algoritmo Diffie-Hellman per lo scambio di chiavi, 83
attacchi man-in-the-middle, 86
certificati, 89
chiave di autenticazione di Oakley, 89-91
chiavi private, 79
chiavi pubbliche, 78
chiavi segrete, 89
curve ellittiche, 79
digital signature standard (DSS), 87
distribuzione delle chiavi, 89
firma digitale, 79
gestione delle chiavi, 89-91
principi, 76-80
protocolli per lo scambio delle chiavi, 85

requisiti della crittografia a chiave pubblica RSA, 80
sistemi di, 79
struttura della cifratura, 76-79
testo cifrato, 77
resto in chiaro, 77
crittografia a chiave pubblica e autenticazione dei messaggi
lettura consigliate, 91
siti web, 91
crittografia delle curve ellittiche
livello di sicurezza, 88
curve ellittiche,
crittografia a chiave pubblica, 79

D

data encryption standard (DES), 35-37
algoritmo (DEA), descrizione, 36
robustezza di, 36
dati firmati, S/MIME, 167
dati in chiaro firmati, S/MIME, 167
DDoS, tipi di attacco, 367
tipi di attacco, inondazione, 367
dispositivi di cifratura,
livello di sicurezza, 53
distribuzione delle chiavi, 54-56
centro per la, 55
chiave di sessione, 55
chiave permanente, 55
modulo del servizio di sicurezza (SSM, *security service module*), 55

E

elaborazioni crittografiche, 247-248
elaborazione del messaggio, SNMPv3, 284
modello, 289
modello di sicurezza utente (USM, *user security model*), 281
ElGamal, 142
encapsulated security payload (ESP), 189-195, 200-209
algoritmi di autenticazione, 190
algoritmi di cifratura, 201

completamento, 202
definizione, 189
formato, 200
informazioni, 192
intestazione, 227
IPSec, 188-191
modalità trasporto, 193-194, 206
modalità tunnel, 194-195
opzione di autenticazione, 206
EnvelopedData, S/MIME, 170-171
esercizi di laboratorio,
sicurezza di rete, 402
ESP, campo dati di autenticazione di, 202, 205-206
Etichette di sicurezza, S/MIME, 176

F

Firewall
bastion host, 381
dual-homed bastion, 382
screened host firewall, 382
single-homed bastion, 382
configurazione, 377, 383
contraffazione degli indirizzi IP (*IP spoofing*), 378
controllo
dei servizi, 372
del comportamento, 372
dell'utente, 372
della direzione, 372
criteri comuni (CC), 390-394
gateway
a livello applicazione, 379
a livello circuito, 380
lettura consigliate e siti web, 395
principi di progettazione, 371-384
router a filtraggio di pacchetti, 373
campo protocollo IP, 374
indirizzi di livello di trasporto, 374
indirizzo IP di destinazione, 374
indirizzo IP sorgente, 373
interfaccia, 374
sistemi trusted, 384-390
siti web, 395
tipologie, 373
a controlli con memoria dello stato (*statefull inspection firewall*), 378

firma
 generalmente fidato (*usually trusted*), 156
 massima fiducia (*ultimate trust*), 156
 sempre fidato (*always trusted*), 156
 valore di fiducia della, 156

Flag
 ticket, Kerberos, 116-118

formato per lo scambio delle informazioni di rilevazione delle intrusioni, IETF, 329

funzione hash
 algoritmo digest del messaggio, MD5, 73
 altre funzioni hash sicure, 72-73
 debole, 67
 e controllo di ridondanza longitudinale, 67
 forte, 67
 funzione di compressione, HMAC, 73-76
 algoritmo, 75
 obiettivi di progetto, 73-76
 requisiti, 66
 resistenza alle collisioni
 debole, 66
 forte, 66
 SHA-1, funzioni sicure, 69
 aggiunta della lunghezza, 70
 aggiunta di bit di completamento, 70
 elaborazione del messaggio, 72
 inizializzazione del buffer hash, 71
 semplici, 67
 sicure e HMAC, 66-76
 unidirezionale, 64
 Whirlpool, 73

G

gauge, rilevazione delle intrusioni, 321
 gestione delle chiavi, tipi di, 209
 gestione delle password, protezione delle, 330
 gestione di rete
 livello di sicurezza, 286
 gestione di rete distribuita, mancanza di supporto, 275
 GetResponse, tipi di messaggio mediante, 272, 273

I

identificatore del protocollo di sicurezza, IPSec, 191
 indice dei parametri di sicurezza (SPI, *security parameters index*), IPSec, 191
 infrastruttura per la chiave pubblica (PKI)
 funzioni di gestione PKIX, 130
 protocolli di gestione PKIX, 131
 intervallo temporale, rilevazione delle intrusioni, 322
 intestazione successiva, ESP, 229
 intrusi, come minaccia alla sicurezza, 313
 intrusione, tecniche di, 315
 intrusioni, 313-317
 base-rate fallacy, 313-317
 cifratura unidirezionale, 315
 Computer Emergency Response Team (CERT), 315
 controllo dell'accesso, 315
 formato per lo scambio delle informazioni di rilevazione delle intrusioni, IETF, 329
 gestione delle password, 330-340
 honeypots, 329
 letture consigliate e siti web, 340
 richieste sospette di invocazione di procedura remota, 314
 rilevazione delle, 317-330
 siti web, 340
 utente
 clandestino, 314
 malintenzionato, 313
 masquerade, 313
 iterazione, cifratura AES, 40

J

Java e ActiveX, minacce a, 361, 362

K

Kerberos, 99-118
 affidabilità, 101
 attacchi alle password, 114
 autenticazione inter-realm, 113
 chiavi di sessione, 114

cifratura pcbc, 114
 differenze tra la versione 4 e 5, 112
 dipendenza dal protocollo Internet, 113
 dipendenza dal sistema di cifratura, 113
 doppia cifratura, 114
 flag del ticket, 116-118
 multiplo, 110-112
 ordinamento dei byte del messaggio, 113
 realm di, 110-112
 scalabilità, 101
 sicurezza, 101
 tempo di validità del ticket, 113
 tipologie di minacce, 99-100
 trasparenza, 101
 versione 4-5, 101, 112
 dialogo di autenticazione, 114-116
 nonce, 115
 numero di sequenza, 116
 opzioni, 115
 realm, 115
 scambi, 116
 sottochiave, 116
 tempi, 115

L

limiti
 di sicurezza, 275
 funzionali, 275
 lunghezza del completamento, ESP, 200

M

mailing list, S/MIME, 176
 man-in-the-middle, attacco, 86, 210, 267
 messaggio server_done, 245
 MIME, tipi di contenuto di, 161-162
 MIME, *si veda multipurpose Internet mail extensions (MIME)*
 minacce
 alla sicurezza, 232, 313
 delle reti, 52
 intrusi, 313
 web, 232
 allo schema di gestione delle password, 332

associate all'autenticazione dell'utente, 99-100
 di servizio, 18
 sicurezza utente, 292
 web, 233
 tipologie di minacce, 233
 worm, 357
 modalità trasporto, ESP, 193-194, 206
 modalità tunnel, ESP, 194-195
 modello di controllo dell'accesso basato su viste (VACM), 302
 elaborazione del controllo degli accessi, 304
 elementi del modello, 302
 in SNMPv3, uso di, 302
 istanza dell'oggetto, 304
 livello di sicurezza, 303
 modello di sicurezza, 286
 motivazione per, 301
 principal che effettua la richiesta, 304
 vista MIB, 280, 303
 multipurpose Internet mail extensions (MIME)
 codifiche di trasferimento, 164-165
 descrizione generale, 160-161
 esempio di messaggio multipart, 163
 forma canonica, 165
si veda anche secure/multipurpose Internet mail extension (S/MIME)
 tipi di contenuto, 161-164
 tipo multipart, 161
 tipo text, 161

N

nonce Kerberos, 115
 notifica,
 applicazione di generazione, SNMPv3, 290
 applicazione di ricezione, SNMPv3, 290
 numero di iterazioni, funzione, cifrario di Feistel, 34
 numero di sequenza, Kerberos, 116

P

password, minacce allo schema di gestione delle, 32

payload, *si veda anche encapsulated security payload (ESP)*, 206

PDU SNMPv1, 277

pretty good privacy (PGP), 137-158

autenticazione, 142

chiavi crittografiche, 139, 144

compatibilità con la posta elettronica, 143

componente firma, 147

componente messaggio, 147

compressione, 142

descrizione operativa, 139

ElGamal, 142

generazione delle chiavi di sessione, 146

gestione delle chiavi pubbliche, 153-158

identificativi delle chiavi, 146

keyring, 144, 149-152

messaggi, trasmissione e ricezione dei, 145

modalità cipher feedback (CFB), 141, 146

notazione, 138

revoca delle chiavi pubbliche, 142, 158

riassembaggio, 144

riservatezza, 141

segmentazione, 144

tipi di chiavi, 145

PRF

livello di sicurezza della funzione, 250

processo di scansione

strategie, 369

casuale, 369

hit-list, 369

sottorete locale, 369

topologica, 369

progetti

di programmazione, sicurezza di rete, 402

di ricerca, sicurezza di rete, 401

per l'insegnamento, sicurezza di rete, 401-402

programmi dolosi, 347-353

auto-roote, 348

backdoor (trapdoor), 348

bomba logica, 348, 349

cavalli di Troia (trojan horse), 348, 349

exploit, 348

flooders (inondatori), 348

keyloggers (registratori di chiavi), 348

rootkit, 348

scaricatori, 348

spam, 348, 358

toolkit (generatore di virus), 348, 354

virus, 348

worm, 348, 356

zombie, 348, 350

protocollo, gestione PKIX, 131

si veda anche protocollo internet (IP); simple network management protocol (SNMP)

protocollo

alert, bad_record_mac, 239

decompression_failure, 239

handshake_failure, 239

lunghezza compressa, 239

unexpected_message, 239

illegal_parameter, 239

no_certificate, 240

bad_certificate, 240

certificate_revoked, 240

certificate_unknown, 240

unsupported_certificate, 240

protocollo handshake di SSL, tipi di messaggio del, 241

protocollo record di SSL, compressione, 237

frammentazione, 237

integrità dei messaggi, 236

lunghezza compressa, 239

riservatezza, 236

tipo di contenuto, 239

versione minore, 239

versione principale, 239

proxy, applicazione per l'inoltro, SNMPv3, 290

SNMPv1, 273

R

realm, Kerberos, 115

requisito MUST, S/MIME, 167

requisito SHOULD, S/MIME, 167

ricevute firmate, S/MIME, 176

richiesta di registrazione, S/MIME, 173

rilevazione delle intrusioni, 317-330

approccio multivariato, 322

basata su regole, 319, 324-326

base-rate fallacy, 313-317

distribuita, 326

formato per lo scambio delle informazioni di rilevazione delle intrusioni, IETF, 329

honeypot, 329

identificazione delle penetrazioni, 325

media e deviazione standard, 322

modello delle serie temporali, 322

modello di Markov, 337, 337, 338

modello operazionale, 322

record di audit, 319-321

record di audit dell'host (HAR), 328, 329

rilevazione basata su regole, 319, 324-326

rilevazione distribuita delle intrusioni, 326

rilevazione statistica delle anomalie, 318, 321-324

rilevazione distribuita delle intrusioni, 326

RSA, cifratura a chiave pubblica, 80

protocolli per lo scambio delle chiavi, 85-86

S

S/MIME, Mailing list sicure, 176

ricevute firmate, 176

tipi di contenuti, 170

S/MIME, *si veda secure/multipurpose Internet mail extension (S/MIME)*

scambi, Kerberos, 116

scambi ISAKMP, tipi di, 218, 219

tipi di, 218, 219

scambio delle chiavi, RSA, 85-86

scambio di chiavi di Diffie-Hellman, 83

algoritmo, 83

tracchi man-in-the-middle, 86

secure electronic transaction (SET), caratteristiche rilevanti, 255

descrizione generale, 254

doppia firma, 258-265

partecipanti, 255-257

secure socket layer (SSL), architettura, 234-236

protocollo alert, 239-241

protocollo change cipher spec, 239

protocollo handshake, 241-247

protocollo record, 236-239

transport layer security (TLS), 248

secure/multipurpose Internet mail extension (S/MIME), 159-176

algoritmi di crittografia, 167-169

basic encoding rules (BER), 170

certificati VerSign, 174-176

clear signing, 172-173

elaborazione dei certificati, 173-176

EnvelopedData, 170-171

etichette di sicurezza, 176

funzionalità, 165-169

mailing list sicure, 176

messaggi, 169-173

messaggio certificate-only, 173

multipurpose Internet mail extensions (MIME), 160-165

rendere sicura un'entità MIME, 169-170

RFC 822, 159-160

ricevute firmate, 176

richiesta di registrazione, 173

ruolo dell'utente, 173-174

servizi di sicurezza potenziati, 176

SignedData, 171-172

Servizio X.509,

attributi del certificato, 128

autenticazione, 119-128

bidirezionale, 126

tridirezionale, 126

unidirezionale, 125

certificati, 120

di tipo forward, 123

di tipo reverse, 123

estensioni, 120

firma, 120

identificatore dell'algoritmo di firma, 120

identificatore unico, 120

informazioni sulla chiave pubblica, 120

nome del soggetto, 120

nome di chi emette, 120

numero di serie, 120

periodo di validità, 120

versione, 120

certificato utente, ottenere il, 122

corrispondenze tra politiche, 128

identificatore della chiave del soggetto, 127

identificatore della chiave dell'autorità, 127

informazioni sulle chiavi, 127-128

informazioni sulle politiche, 127-128

periodo di utilizzo della chiave, 127

politiche riguardanti i certificati, 128

procedure di autenticazione, 124
 revoca dei certificati, 124
 sul percorso di certificazione, 128
 utilizzo della chiave, 127
 versione 3, 126
SET, tipi di transazione, 260
SET, *si veda secure electronic transaction (SET)*
 sicurezza dei sistemi di elaborazione,
 definizione, 1
 sicurezza della gestione di rete, 269-310
 modello di controllo dell'accesso basato su viste
 (VACM), 302
 SNMPv3, 281-306
 sicurezza della posta elettronica (e-mail),
 137-176
 siti web, 176-177
 sicurezza di rete
 compiti di lettura/relazione, 402
 compiti scritti, 402
 esercizi di laboratorio, 402
 progetti di programmazione, 402
 progetti di ricerca, 401
 progetti per l'insegnamento, 401-402
 sicurezza IP (IPSec), 185-230
 algoritmi di cifratura e autenticazione, 201-202
 applicazioni, 185-186
 applicazioni di instradamento, 188
 architettura, 188-194
 associazioni di sicurezza (SA), 191-193
 autenticazione, 206-209
 bundle trasporto-tunnel, 207
 caratteristiche di Oakley, 211-213
 combinazione di più SA, 206
 combinazioni base di SA, 207-209
 descrizione generale, 185-188
 documenti, 188-190
 encapsulated security payload (ESP), 200-206
 con opzione di autenticazione, 206-207
 in modalità trasporto, 202-204
 in modalità tunnel, 205
 esempio di scambi mediante Oakley, 213-214
 formato dell'intestazione ISAKMP, 214-215
 formato di ESP, 200-201
 gestione delle chiavi, 209-220
Internet security and key management protocol (ISAKMP), 214-220
 internetworking e protocolli Internet, 222-230
 intestazione di autenticazione (AH), 195-200

IPv4, 224-226
 IPv6, 226-230
ISAKMP, 214-220
 letture consigliate e siti web, 220
 modalità trasporto, 193-194
 modalità trasporto e tunnel, 198-200
 modalità tunnel, 194-194
 parametri, 191-192
 protocollo Oakley per la scelta
 delle chiavi, 210
 riservatezza, 206-209
 scambi ISAKMP, 218-220
 selettori, 192-193
 servizi, 190-191
 servizio anti-replay, 196-197
 siti web, 220
 tipologie di payload ISAKMP, 215-218
 transport adjacency, 207
 valore per la verifica di integrità, 197-198
 vantaggi, 186-188
 sicurezza utente, minacce, 292
 sicurezza web, 231-267
 approcci alla sicurezza del traffico, 232-234
 considerazioni, 231-234
 letture consigliate e siti web, 231-266
 minacce, 232-234
 secure electronic transaction (SET), 253-265
 secure socket layer (SSL) e transport layer security (TLS), 4-253
 siti web, 231-266
SignedData, S/MIME, 171-172
 sistemi trusted, 384-390
 capability ticket, 386
 concetto di, 386
 controllo dell'accesso ai dati, 384
 difesa dai cavalli di Troia, 388
 diritto d'accesso, 385
 lettura verso l'alto, 386
 proprietà Star (*), 386
 scrittura verso il basso, 386
 simple security property, 386
 strutture di controllo dell'accesso, 385
 siti web, autenticazione dei messaggi, 91
SNMP
 applicazioni di generazione delle notifiche, 285
 applicazioni di risposta ai comandi, 285
 applicazioni proxy per l'inoltro, 285

architettura di gestione di rete, 270
 concetti base, 270-278
SNMPv1
 amministrazione di, 280
 comando GET, 278
 comunità, 278
 comunità SNMP, 279
 funzionalità di comunità di, 278-280
 livello di sicurezza, 278
 servizio di autenticazione, 279
 tipi di limiti, 275
SNMPv3, 281-305
 agente SNMP tradizionale, 285
 applicazione di generazione dei comandi, 289
 applicazione di risposta ai comandi, 289
 applicazione proxy per l'inoltro, 290
 applicazioni, 287-290
 architettura, 282
 entità SNMP, 282
 gestore SNMP tradizionale, 283
 modello di controllo dell'accesso basato su viste
 (VACM), 302
 scoped PDU, 291
 terminologia, 286
 user security model (USM), 281
 software dannosi
 backdoor, 348
 bomba logica, 348, 349
 software dolosi
 lettura consigliate e siti web, 369
 struttura del cifrario di Feistel, 33-35
SunOS
 tipi di eventi, 326

T

teoria dei numeri, 397-400
 aritmetica modulare, 399
 divisori, 397
 massimo comune divisore, 398
 numeri primi, 397, 398
 numeri relativamente primi, 397,
 398
 residuo, 399
 tipi di attacchi, 32
 tipi di cifratura, 29

tipi di PDU
 inform, 209
 trap, 209
 tipi di RFC, 22
 tipologie di attacchi effettuabili su messaggi
 cifrati, 32
transport layer security (TLS)
 change_cipher_spec, 247
 codice di autenticazione del messaggio (MAC),
 248-253
transport layer security (TLS), codici
 di allarme, 251
 completamento, 253
 funzione pseudocasuale (PRF), 249
 messaggio finished, 247
 suite di cifratura, 252
 trasposizione, 30

U

user security model (USM), Modello
 di sicurezza utente, in SNMPv3 uso del, 281
USTAT, 326
 utilizzo delle risorse, rilevazione
 delle intrusioni, 317

V

valore
 del campo di legittimità della chiave, 156
 di fiducia della firma, 156
 di massima fiducia (*ultimate trust*), 154
 segreto principale (*master secret*), SSL, 236
 veicoli di trasporto, worm, 359
 virus
 del settore di boot, 354
 della posta elettronica, 355
 delle macro, 355
 furtivo, 354
 infezione iniziale, 353
 metamorfici, 354
 minacce correlate, 347-359
 natura dei, 350
 parassiti, 354
 polimorfi, 354

residenti in memoria, 354
timedi, 359-364
struttura dei, 351
tipologie di, 353

W

Web, tipologie di minacce, 233

worm

azione del giorno zero, 359
Code Red, 357
Code Red II, 358
veicoli di trasporto, 359
a diffusione ultraveloce, 358
attacchi recenti tramite, 357
conessioni di rete e, 356, 357, 363
definizione, 356

Morris, 357
multi piattaforma, 358
multi-exploit, 358
mydoom, 358
polimorfi, 358
SQL slammer, 358
stato dell'arte della tecnologia degli, 358

Z

ZIP

algoritmo
di compressione, 179-180
di decompressione, 180
LZ77, 178
zombie, 348, 350