CAPITOLO

8

Sicurezza nelle reti

Ricorderete che nel Paragrafo 1.6 abbiamo passato in rassegna numerose tipologie di attacco, comprese diverse forme di attacchi DoS, malware, analisi del traffico, mascheramento della sorgente e modifica e cancellazione di messaggi. Avendo ora acquisito una buona conoscenza di Internet e dei suoi protocolli siamo pronti a studiare in dettaglio la comunicazione sicura e le tecniche di difesa della rete.

In questo capitolo faremo la conoscenza di Alice e Bob, due persone che vogliono comunicare e desiderano farlo in modo sicuro. E dato che questo testo si occupa di reti, è opportuno sottolineare che Alice e Bob potrebbero essere due router che vogliono scambiarsi tabelle di instradamento in modo sicuro, un client e un server che desiderano stabilire una connessione di trasporto sicura o due applicazioni di posta elettronica che vogliono scambiarsi e-mail sicure: tutti casi concreti che considereremo in seguito in questo capitolo. Alice e Bob sono due figure ben note nella comunità della sicurezza, forse perché usare i loro nomi è più simpatico del ricorso a locuzioni come: *la generica entità chiamata "A" vuole scambiare dati in modalità sicura con una specifica entità detta "B"*. Relazioni sentimentali, trasmissione di messaggi criptati in tempo di guerra e le transazioni commerciali sono le più comuni circostanze in cui è richiesta una comunicazione sicura. Preferendo la prima alle ultime due, useremo alternativamente Alice e Bob come mittente e ricevente nel caso di una corrispondenza fra due innamorati.

Ma, precisamente, che cosa significa "comunicare in modo sicuro"? Nel nostro esempio possiamo immaginare che Alice e Bob vogliano avere la certezza che stanno effettivamente parlando tra loro, che le loro conversazioni rimangano segrete, protette da ogni possibile intrusione e che l'intercettazione dei messaggi sia immediatamente scoperta. Nella prima parte del capitolo analizzeremo le tecniche fondamentali che permettono la cifratura e decifratura delle comunicazioni, l'autenticazione della persona con cui si comunica e la prova dell'integrità del messaggio.

Nella seconda parte del capitolo esamineremo come i princìpi fondamentali della cifratura possono essere usati per creare protocolli di rete sicuri. Ancora una volta, adottando un approccio top-down, studieremo i protocolli sicuri in ciascuno dei quattro livelli in cima alla pila di protocolli, iniziando dal livello di applicazione. Vedremo come rendere sicura la posta elettronica, una connessione TCP, come fornire sicurezza a livello di rete e come rendere sicura una LAN wireless.

Nella terza parte di questo capitolo considereremo la sicurezza dal punto di vista operativo, che consente di proteggere una rete istituzionale dagli attacchi. In particolare esamineremo con attenzione come i firewall e i sistemi di rilevamento delle intrusioni possono aumentare la sicurezza di una rete istituzionale.

8.1 Sicurezza di rete

Iniziamo il nostro studio sulla sicurezza delle reti specificando che cosa significa esattamente "comunicare in modo sicuro". Facendo riferimento ai nostri personaggi, possiamo immaginare che Alice voglia che solo Bob sia in grado di comprendere il messaggio inviato, anche se stanno utilizzando un canale di trasmissione non sicuro, per cui un intruso (una certa Trudy) potrebbe intercettare qualunque cosa trasmessa tra Alice e Bob. D'altra parte, Bob vuole a sua volta essere certo che quanto ricevuto provenga veramente da Alice e, parimenti, Alice vuole essere sicura che la persona con cui sta comunicando sia veramente Bob. Entrambi vogliono che, in ogni caso, la loro corrispondenza non subisca alterazioni durante il trasporto. Ma, soprattutto, vogliono poter comunicare, ossia che nessuno neghi loro l'accesso alle risorse necessarie per lo scambio di informazioni. Fatte queste considerazioni, possiamo identificare le proprietà auspicabili per la **comunicazione sicura**.

- *Riservatezza*. Solo mittente e destinatario dovrebbero essere in grado di comprendere il contenuto del messaggio trasmesso. Dato che questo può essere intercettato da qualche "spione",[1] è necessario *cifrarlo* in modo da renderlo incomprensibile a chi lo intercetta. La segretezza è forse l'aspetto più comunemente associato alla sicurezza. Studieremo in seguito le tecniche crittografiche per cifrare e decifrare i dati (Paragrafo 8.2).

[1] Traduciamo così il termine inglese "eavesdropper" con cui viene indicata una persona che spia una conversazione in maniera illecita (*N.d.R.*).

- *Integrità del messaggio*. Oltre all'autenticazione di mittente e destinatario, occorre che il contenuto della comunicazione non subisca, durante la trasmissione, alterazioni dovute a cause fortuite o a manipolazioni. Per garantire l'integrità dei messaggi si possono utilizzare estensioni delle tecniche di checksum, analizzate nei protocolli di trasferimento affidabile e di collegamento (Paragrafo 8.3).
- *Autenticazione*. Mittente e destinatario devono essere reciprocamente sicuri della loro identità, cioè devono poter confermare che l'altra parte sia effettivamente chi dichiara di essere. Indubbiamente, in un contatto diretto, *de visu*, questo problema non si pone; non altrettanto avviene con la comunicazione a distanza. Per esempio, come può un server e-mail avere la certezza che la persona che accede a una cartella di posta sia quella autorizzata? Esamineremo nel Paragrafo 8.4 le tecniche di autenticazione.
- *Sicurezza operativa*. Quasi tutte le istituzioni (aziende, università e così via) hanno reti collegate a Internet, che potrebbero essere compromesse da attaccanti che riescono a entrare in esse tramite Internet. Gli attaccanti possono tentare di installare dei worm negli host della rete, ottenere segreti commerciali, tracciare la configurazione interna della rete e lanciare attacchi DoS. Vedremo nel Paragrafo 8.9 che dispositivi operativi, come firewall e sistemi di rilevamento delle intrusioni, sono usati per contrastare gli attacchi contro le reti delle istituzioni. Un firewall si pone tra la rete dell'istituzione e quella pubblica e controlla i pacchetti in transito. Un sistema di rilevamento delle intrusioni esegue un controllo approfondito dei pacchetti, avvisando gli amministratori di rete in caso di attività sospette.

Consideriamo ora quali sono le azioni che un intruso può intraprendere e a quale tipo di informazioni può accedere. La Figura 8.1 illustra uno scenario in cui Alice vuole inviare un'e-mail a Bob. Per comunicare in modo sicuro, rispettando i requisiti di riservatezza, autenticazione e integrità di messaggio, i due si scambieranno pacchetti di controllo e di dati, in modo similare a quanto avviene con TCP, e tutti o alcuni di questi messaggi saranno tipicamente cifrati. A questo punto un malintenzionato potrebbe, come abbiamo visto nel Paragrafo 1.6, procedere con una delle seguenti azioni.

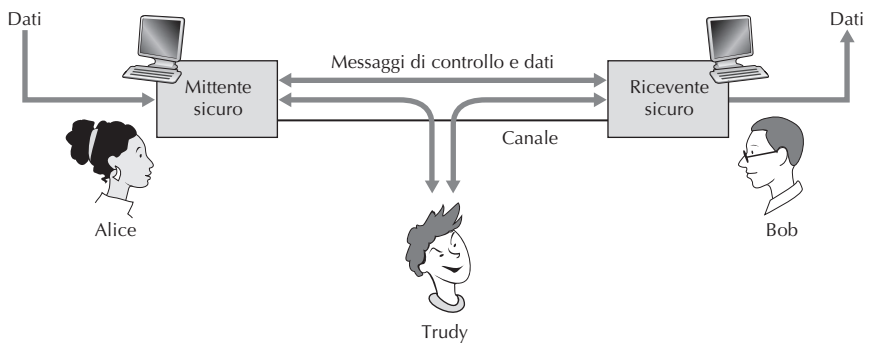

Figura 8.1 Mittente, ricevente e intruso (Alice, Bob e Trudy).

- *Ascoltare e registrare i messaggi di controllo e dati in transito sul canale* (intercettazione o eavesdropping).
- *Rimuovere, aggiungere o modificare i messaggi o il loro contenuto*.

Se non si adottano appropriate contromisure, l'intruso ha quindi la possibilità di mettere in atto un'ampia gamma di attacchi: spiare la conversazione (eventualmente rubando password e dati), impersonare un altro soggetto, dirottare (*hijack*) una sessione in corso, negare il servizio agli utenti legittimi, sovraccaricando le risorse del sistema, e così via. Un riassunto degli attacchi denunciati viene mantenuto dal Centro di Coordinamento del CERT [CERT 2016].

Stabilito che esistono reali minacce annidate in Internet, quali potrebbero essere le vittime di potenziali aggressioni? Potrebbe, per esempio, essere un consumatore che trasmette il suo numero di carta di credito a un web server per effettuare un acquisto. Oppure, il cliente che interagisce on-line con la propria banca. I soggetti che richiedono comunicazione sicura potrebbero anche essere parte integrante dell'infrastruttura di rete. Ricordiamo che il DNS (Paragrafo 2.4) e i sistemi che si scambiano le tabelle di instradamento (Capitolo 5) richiedono la sicurezza della comunicazione fra due parti. Lo stesso vale per le applicazioni di gestione della rete (Capitolo 5). Un malintenzionato che può attivamente interferire, controllare o danneggiare la ricerca e l'aggiornamento di un DNS, il calcolo dell'instradamento [RFC 4272] o le funzioni di gestione della rete [RFC 3414] è in grado di provocare seri danni a Internet.

Avendo stabilito l'infrastruttura, alcune delle definizioni più importanti e la necessità della sicurezza di rete, parliamo ora della crittografia. La crittografia si pone come pietra angolare della sicurezza di rete e il suo utilizzo risulta fondamentale non solo per fornire riservatezza, ma anche come un importante strumento per il servizio di autenticazione e l'integrità del messaggio.

8.2 Princìpi di crittografia

Anche se alcuni fanno risalire le sue origini ai tempi di Giulio Cesare, la crittografia moderna basa le sue tecniche sugli sviluppi verificatisi negli ultimi trent'anni. Certo, una completa trattazione di questo argomento richiederebbe un intero volume, e qui ne affronteremo solo gli aspetti essenziali, soprattutto in rapporto alla loro applicazione a Internet. *The Codebreakers* [Kahn 1967] e *The Code Book*: *The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh 1999] forniscono una panoramica affascinante di questa lunga storia. Una dettagliata e brillante esposizione tecnica della crittografia, soprattutto dal punto di vista della rete, si trova in [Kaufman 1995]. [Diffie 1998] fornisce un esame appassionante e aggiornato degli aspetti politici e sociali (per esempio, la *privacy*) che sono ora inestricabilmente intrecciati con la crittografia. Una completa trattazione della crittografia si trova in [Kaufman 1995; Schneier 1995]. Sebbene in questo paragrafo ci concentreremo sull'uso della critto-

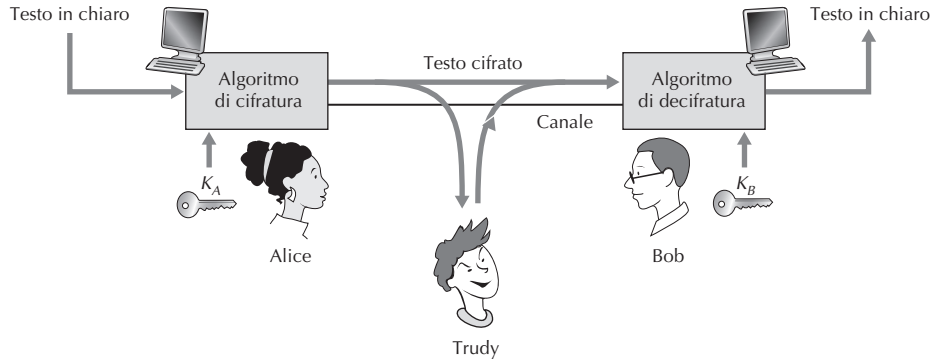

Figura 8.2 Componenti crittografici.

grafia per la riservatezza, vi facciamo notare che è inestricabilmente legata all'autenticazione, all'integrità dei messaggi, al non ripudio e ad altro ancora.

In pratica, le tecniche di crittografia consentono al trasmittente di mascherare i dati in modo che un intruso non possa comprenderne il contenuto. Il ricevente, naturalmente, deve invece essere in grado di recuperare i dati originali. La Figura 8.2 mostra alcuni termini importanti.

Supponete che Alice voglia inviare un messaggio a Bob. Il messaggio originario scritto da Alice (per esempio, "Bob, ti amo. Alice") è detto **testo in chiaro** (*plaintext* o *cleartext*); questo poi viene trasformato da un **algoritmo di cifratura** in un **messaggio cifrato** (*ciphertext*) che risulta inintelligibile a qualsiasi intruso. È importante notare che in molti casi le tecniche di crittografia, comprese quelle utilizzate in Internet, sono *di dominio pubblico*, standardizzate e disponibili a chiunque (per esempio, [RFC 1321, RFC 3447, RFC 2420, NIST 2001]), anche a un potenziale intruso. È evidente che, se tutti conoscono il metodo per cifrare i dati, occorrono informazioni segrete che impediscono ai curiosi malevoli di decifrare i dati trasmessi. Qui entrano in gioco le chiavi.

Nella Figura 8.2 Alice fornisce all'algoritmo di cifratura una **chiave**, K_A, cioè una stringa alfanumerica che genera il testo cifrato. La notazione $K_A(m)$ denota la forma cifrata utilizzando la chiave K_A del messaggio in chiaro, m. Lo specifico algoritmo di cifratura che utilizza la chiave K_A risulterà evidente dal contesto. Analogamente, Bob fornisce una chiave, K_B, all'**algoritmo di decifratura** che legge il testo cifrato e restituisce quello originale in chiaro. Cioè, quando Bob riceve un messaggio cifrato $K_A(m)$, lo può decifrare calcolando $K_B(K_A(m)) = m$. Nei cosiddetti **sistemi a chiave simmetrica** le chiavi di Alice e Bob sono identiche e segrete. Nei **sistemi a chiave pubblica** si utilizzano due chiavi. Una è di pubblico dominio, l'altra è conosciuta o da Bob o da Alice, ma non da entrambi. Nei due paragrafi seguenti considereremo i sistemi a chiave simmetrica e a chiave pubblica in maggiore dettaglio.

8.2.1 Crittografia a chiave simmetrica

Gli algoritmi di cifratura comportano la sostituzione del messaggio in chiaro con uno codificato in modo che risulti inintelligibile a chi non conosca la chiave. A titolo esemplificativo soffermiamoci su un antico algoritmo a chiave simmetrica conosciuto come **cifrario di Cesare**.

Il suo funzionamento è tanto semplice quanto efficace: si procede alla sostituzione di ciascuna lettera del messaggio in chiaro con un'altra sfasata, rispetto alla prima, di un numero k di posti nell'alfabeto. Per esempio, se $k = 3$, allora la lettera "a" diventa "d", la "c" diventa "f", e così via, facendo procedere lo scambio in senso ciclico, vale a dire che una volta terminato l'alfabeto si ricomincia con la lettera "a". In questo caso la chiave è costituita dal valore di k.

Quindi, il messaggio "bob, ti amo, alice." diventerebbe "ere, wl dpr. dolfh". La frase, all'apparenza completamente assurda, può essere facilmente decifrata a patto di sapere che è stato impiegato il cifrario di Cesare: i possibili valori della chiave sono, per l'alfabeto inglese, 25.

Uno sviluppo rispetto al metodo appena esposto è costituito dal cosiddetto **cifrario monoalfabetico**. Anche in questo caso si procede alla scambio di una lettera dell'alfabeto con un'altra, la sostituzione non avviene però seguendo uno schema regolare, ma cambiando tutte le occorrenze di una data lettera con un'altra – sempre la stessa – scelta in modo arbitrario (Figura 8.3).

In questo caso il messaggio "bob, ti amo, Alice." diventerebbe "nkn, us mhk. Mgsbc". Il **cifrario monoalfabetico** appare più efficiente di quello di Cesare, in quanto usando un alfabeto di 26 lettere (come quello inglese) ci troviamo di fronte a ben 26 fattoriale differenti possibilità di accoppiamento (un valore dell'ordine di 10^{26}) anziché le 25 di prima. La ricerca dei giusti abbinamenti con un approccio brutale, come provare tutti i 10^{26} accoppiamenti richiederebbe un tempo eccessivo per essere praticabile. Tuttavia, basandosi su informazioni statistiche, come il fatto che le lettere "e" e "t" sono quelle che ricorrono con maggiore frequenza nelle parole inglesi (valgono il 13% e il 9% delle occorrenze) o che alcuni gruppi di lettere appaiono spesso insieme (in italiano gruppi come "che", "zione" e "mente" sono molto comuni), è possibile rendere più agevole la violazione del codice. Ancor più se l'intruso ha alcune cognizioni sul possibile contenuto del messaggio. Per esempio, se Trudy, la moglie di Bob, fosse a conoscenza della sua relazione con Alice, allora potrebbe supporre che i loro nomi compaiano nel testo cifrato e determinare nove dei 26 accoppiamenti di lettere e riducendo di 10^9 le combinazioni da controllare. Inoltre, se Trudy sospetta che Bob abbia una relazione con Alice, si aspetterà di trovare altre parole note nel messaggio.

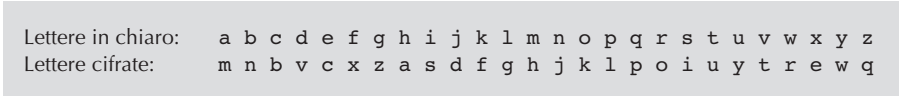

Figura 8.3 Cifrario monoalfabetico.

In generale, si possono immaginare tre differenti scenari di decodifica, in funzione delle informazioni possedute dall'intruso.

- *Attacco al testo cifrato*. È il caso in cui l'intruso ha solamente accesso al testo intercettato senza alcuna indicazione sul contenuto del messaggio e può avere come unico supporto l'analisi statistica.
- *Attacco con testo in chiaro noto*. Si verifica quando l'intruso conosce alcuni possibili accoppiamenti (testo in chiaro/testo cifrato). Nel caso di Trudy, abbiamo visto che se sa che i nomi dei due corrispondenti compaiono nel messaggio cifrato, può facilmente determinare le lettere che sostituiscono *a, b, c, e, i, l, o*. Oppure potrebbe disporre di altri supporti come la registrazione delle precedenti trasmissioni o aver sottratto la minuta di un messaggio scritto "in chiaro" da Alice.
- *Attacco con testo in chiaro scelto*. Costituisce il caso in cui l'intruso è in grado di ottenere la forma cifrata di un messaggio a lui noto. Con i semplici algoritmi di cifratura visti in precedenza, se Trudy riesce a intercettare un messaggio di Alice che contiene tutte le lettere dell'alfabeto, allora può decifrare lo schema crittografico. Vedremo tra poco che, con tecniche più sofisticate, un attacco con testo in chiaro scelto non comporta necessariamente la violazione del cifrario.

Ideata cinquecento anni fa, la **cifratura polialfabetica** rappresenta un'ulteriore evoluzione delle precedenti forme di crittografia attraverso l'impiego di molteplici sostituzioni monoalfabetiche. In sostanza, le varie occorrenze di una stessa lettera vengono codificate in modo diverso a seconda della posizione in cui appaiono nel messaggio in chiaro. La Figura 8.4 mostra uno schema di cifratura polialfabetica con due diversi cifrari di Cesare, C_1 con $k = 5$ e C_2 con $k = 19$ che potremmo scegliere di utilizzare seguendo la sequenza C_1, C_2, C_2, C_1, C_2. Cioè, la prima lettera del testo in chiaro deve essere sostituita con C_1, la seconda e la terza con C_2, la quarta con C_1 e la quinta con C_2. Lo schema si ripete in modo ciclico, per cui la sesta lettera verrà scambiata con C_1, e così via. La versione cifrata di "Bob, ti amo." diventerà quindi "ghu, mn trh" Notiamo che la prima "b" di "Bob" è cifrata usando C_1 mentre la seconda usando C_2. In questo esempio la "chiave" di cifratura e decifratura è costituita dalla conoscenza sia delle due chiavi di Cesare ($k = 5$, $k = 19$) sia della sequenza C_1, C_2, C_2, C_1, C_2.

Lettere in chiaro:	a b c d e f g h i j k l m n o p q r s t u v w x y z
$C_1(k = 5)$:	f g h i j k l m n o p q r s t u v w x y z a b c d e
$C_2(k = 19)$:	t u v w x y z a b c d e f g h i j k l m n o p q r s

Figura 8.4 Cifrario polialfabetico ottenuto con due "cifrari di Cesare".

Tabella 8.1 Un esempio di cifrario a blocchi di 3 bit.

Ingresso	Uscita	Ingresso	Uscita
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

Cifrari a blocchi

Esaminiamo ora come viene effettuata la crittografia a chiave simmetrica. Esistono due grandi classi di tecniche di cifratura simmetrica: i **cifrari a flusso** e i **cifrari a blocchi**. In questo paragrafo ci concentriamo sui cifrari a blocchi, che sono usati in molti protocolli sicuri di Internet, compreso PGP (per rendere sicura la posta elettronica), SSL (per rendere sicura una connessione TCP) e IPSec (per rendere sicura la trasmissione a livello di rete).

In un cifrario a blocchi, il messaggio da cifrare è elaborato in blocchi di k bit. Per esempio, se $k = 64$, allora il messaggio viene suddiviso in blocchi di 64 bit e ciascun blocco viene cifrato in modo indipendente. Per codificare un blocco, il cifrario usa una corrispondenza uno a uno per far corrispondere il blocco di k bit di testo in chiaro ai blocchi di k bit di testo cifrato. Consideriamo un semplice esempio per esaminare i codici a blocco in maggior dettaglio. Supponete che $k = 3$ in modo che il cifrario a blocchi faccia corrispondere 3 bit in ingresso (testo in chiaro) a tre bit in uscita (testo cifrato). Una possibile corrispondenza è data dalla Tabella 8.1.

Notate che vi è una corrispondenza uno a uno: cioè vi è un'uscita diversa per ciascun ingresso. Questo cifrario a blocchi suddivide il messaggio in blocchi di 3 bit e cifra ciascun blocco in base alla corrispondenza precedente. Come è possibile verificare, il messaggio 010110001111 viene cifrato ottenendo 101000111001.

Continuando con l'esempio di blocchi a 3 bit, notate che la corrispondenza precedente è solo una delle molte possibili corrispondenze. Quante possibili corrispondenze ci sono? Per rispondere a questa domanda, osservate che una corrispondenza non è nient'altro che una permutazione di tutti i possibili ingressi. Ci sono 2^3 (= 8) possibili ingressi, elencati nella colonna "Ingresso" che possono essere permutati in 8! = 40.320 modi possibili. Dato che ciascuna di queste permutazioni specifica una corrispondenza, ci sono 40.320 possibili corrispondenze. Possiamo vedere ciascuna corrispondenza come una chiave: se Alice e Bob conoscono entrambi la corrispondenza (la chiave) possono cifrare e decifrare i messaggi che si scambiano.

L'attacco a forza bruta per questo cifrario consiste nel provare a decifrare il testo cifrato usando tutte le possibili corrispondenze. Con solo 40.320 corrispondenze (quando $k = 3$) questo può essere svolto velocemente da un PC. Per contrastare un attacco a forza bruta, i cifrari a blocchi tipicamente usano blocchi molto più grandi, con $k = 64$ o maggiore. Notate che il numero di possibili corrispondenze per un ci-

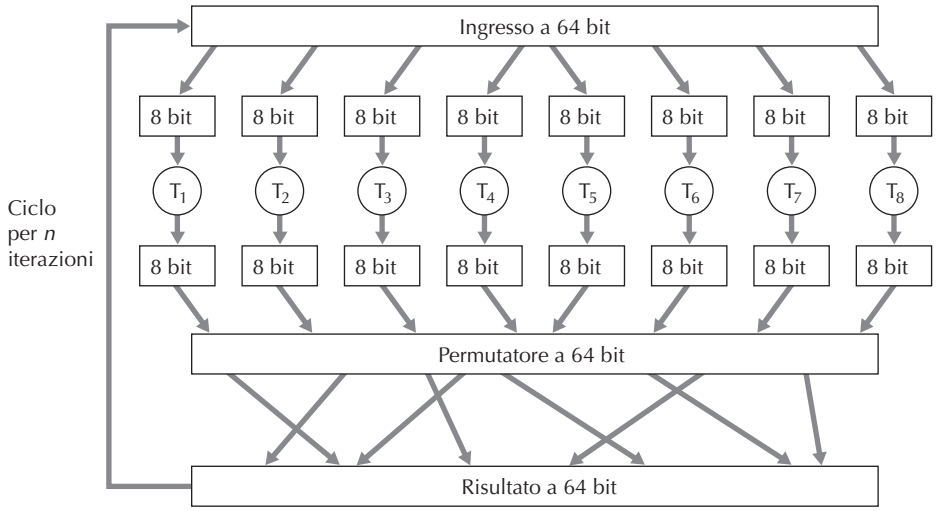

Figura 8.5 Un esempio di cifrario a blocchi.

frario generale con blocchi di k bit è $2^k!$, che è un valore astronomico anche per valori di k non enormi (come $k = 64$).

Sebbene i cifrari a blocchi con una tabella completa, come quelli appena descritti, con valori di k non eccessivamente grandi possono produrre schemi di cifratura a chiave simmetrica robusti, sfortunatamente sono difficili da implementare. Per $k = 64$ e per una certa corrispondenza, Alice e Bob dovrebbero mantenere una tabella con 2^{64} valori in ingresso: un compito improponibile. Inoltre, se Alice e Bob devono cambiare la chiave, dovrebbero entrambi rigenerare la tabella. Quindi, un cifrario a blocchi con la tabella completa che fornisce una corrispondenza pre-determinata tra tutti gli ingressi e le uscite (come nell'esempio precedente) è semplicemente fuori discussione.

Invece, i cifrari a blocchi usano tipicamente funzioni che simulano in modo casuale tabelle permutate. Un esempio (adattato da [Kaufman 1995]) di una funzione di questo tipo, per $k = 64$, è illustrato nella Figura 8.5. La funzione prima suddivide il blocco di 64 bit in 8 parti di 8 bit ciascuna. Ciascuna parte viene elaborata da una tabella di 8 × 8 bit, che ha quindi una dimensione più maneggevole. Per esempio, il primo pezzo è elaborato dalla tabella indicata con T_1. Successivamente le 8 parti in uscita vengono riassemblate nel blocco a 64 bit. Le posizioni dei 64 bit nel blocco vengono poi mescolate (permutate) per produrre l'uscita a 64 bit. Questo risultato viene rinviato all'ingresso a 64 bit, dove inizia un'altra iterazione. Dopo n di queste iterazioni, la funzione fornisce il testo cifrato del blocco a 64 bit. Lo scopo delle iterazioni è quello di far in modo che ciascun ingresso influenzi molti, se non tutti, i bit del risultato finale. Se venisse usata una sola iterazione, un certo bit di ingresso influenzerebbe solo 8 dei 64 bit in uscita. La chiave di questo algoritmo di cifratura a blocchi sarà costituita dalle 8 tabelle di permutazione, supponendo che sia fissa la funzione di mescolamento.

Oggi ci sono parecchi cifrari a blocchi comuni, compreso DES (*data encryption standard*), 3DES, e AES (*advanced encryption standard*). Ciascuno di questi standard usa funzioni, piuttosto che tabelle predeterminate lungo le righe della Figura 8.5 (sebbene più complesse e specifiche per ciascun cifrario). Ciascuno di questi algoritmi usa anche una stringa di bit come chiave. Per esempio DES usa dei blocchi a 64 bit con una chiave a 56 bit. AES usa blocchi di 128 bit e può funzionare con chiavi di 128, 192 o 256 bit. Una chiave di un algoritmo determina le corrispondenze specifiche delle "mini-tabelle" e le permutazioni all'interno dell'algoritmo. L'attacco a forza bruta per ciascuno di questi cifrari consiste nel ripetere le operazioni su tutte le chiavi, applicando l'algoritmo di decifratura con ciascuna chiave. Osservate che con una chiave di lunghezza n, ci sono 2^n possibili chiavi. NIST [NIST 2001] ha stimato che una macchina in grado di ricostruire il DES in un secondo (cioè provare tutte le 2^{56} chiavi in un secondo), ci metterebbe approssimativamente 149 trilioni di anni per decifrare AES con una chiave a 128 bit.

Cifrari a blocchi concatenati

Nelle applicazioni di rete abbiamo sovente bisogno di cifrare lunghi messaggi, o meglio lunghi flussi di dati. Se applicassimo un cifrario a blocchi, descritto precedentemente, semplicemente suddividendo il messaggio in blocchi di k bit e cifrando indipendentemente ciascun blocco, si verificherebbe un problema sottile, ma importante. Per accorgercene, notiamo che due o più blocchi di testo in chiaro possono essere identici, per esempio "HTTP/1.1". Per questi blocchi identici, un cifrario a blocchi produrrebbe, ovviamente, lo stesso testo cifrato. Un attaccante, conoscendo forse il protocollo di rete usato tra Alice e Bob, potrebbe, potenzialmente, indovinare il testo in chiaro, quando vede blocchi di testo cifrato uguali. L'attaccante potrebbe anche essere in grado di decifrare l'intero messaggio, identificando i blocchi di testo cifrato identici e usando le conoscenze sulla struttura dei protocolli sottostanti [Kaufman 1995].

Per risolvere questo problema i cifrari a blocchi usano una tecnica chiamata **cifrari a blocchi concatenati** (*cipher block chaining*, CBC). Per spiegare come funziona, sia $m(i)$ l'i-esimo blocco di testo in chiaro, $c(i)$ l'i-esimo blocco di testo cifrato e $a \oplus b$ sia l'OR esclusivo (XOR) di due stringhe di bit a e b. Indichiamo con K_s anche l'algoritmo di cifratura a blocchi con chiave S. CBC funziona come segue. Il mittente genera una stringa di k bit, $r(i)$ per l'i-esimo blocco e calcola $c(i) = K_s\,(m(i) \oplus r(i))$. Si noti che un nuovo numero viene estratto per ogni blocco. Il mittente invia quindi $c(1)$, $r(1)$, $c(2)$, $r(2)$, $c(3)$, $r(3)$, e così via. Il ricevente, avendo ricevuto $c(i)$ e $r(i)$, può ricostruire ogni blocco calcolando $m(i) = K_s\,(c(i)) \oplus r(i)$. È importante notare che Trudy, sebbene possa spiare $r(i)$ perché inviato in chiaro, non può ottenere $m(i)$, perché non conosce la chiave K_s. Si noti inoltre che anche se due blocchi di testo $m(i)$ e $m(j)$ fossero uguali, i corrispondenti blocchi cifrati sarebbero differenti, in quanto con alta probabilità sarebbero diversi i numeri casuali $r(i)$ e $r(j)$.

Consideriamo come esempio il blocco cifrato a 3 bit della Tabella 8.1. Supponiamo che il testo sia 010010010. Se Alice cifrasse direttamente il testo, senza alcuna

sorgente di casualità, il testo cifrato risultante sarebbe 101101101. Poiché i tre gruppi cifrati sono uguali, se Trudy spiasse il testo cifrato se ne accorgerebbe. Supponiamo invece che Alice generi i blocchi casuali $r(1) = 001$, $r(2) = 111$, e $r(3) = 100$ e usi la tecnica appena spiegata per generare il testo cifrato $c(1) = 100$, $c(2) = 010$ e $c(3) = 000$. I risultanti blocchi cifrati saranno diversi. Alice quindi invia $c(1)$, $r(1)$, $c(2)$ e $r(2)$. Verificate che Bob sia in grado di ottenere il testo originario usando K_s.

Il lettore astuto noterà che l'introduzione di una componente casuale risolve un problema, ma ne introduce un altro. Infatti Alice deve inviare il doppio dei bit, uno cifrato e il corrispondente casuale, raddoppiando la richiesta di banda. Per risolvere questo problema, i cifrari a blocchi usano la già citata tecnica chiamata **cipher block chaining** (CBC). L'idea di base è di inviare un solo numero casuale con il primo messaggio e quindi usare i blocchi calcolati come numero casuale successivo. CBC opera nel seguente modo:

1. Prima di cifrare il messaggio o il flusso di dati, il mittente genera una stringa di k bit, chiamata **vettore di inizializzazione** (IV, *initialization vector*), che indicheremo con $c(0)$. Il mittente manda IV *in chiaro* al destinatario.
2. Per il primo blocco, il mittente calcola $m(1) \oplus c(0)$, cioè l'OR esclusivo tra il primo blocco di testo in chiaro e IV. Poi usa il risultato come input per l'algoritmo di cifratura a blocchi e ottiene il corrispondente blocco di testo cifrato, cioè $c(1) = K_s\,(m(1) \oplus c(0))$. Il mittente manda il blocco di testo cifrato $c(1)$ al ricevente.
3. Per l'i-esimo blocco, il mittente genera l'i-esimo blocco di testo cifrato come, $c(i) = K_s\,(m(i) \oplus c(i-1))$.

Esaminiamo ora alcune conseguenze di tale approccio. In primo luogo il destinatario sarà ancora in grado di recuperare il messaggio originale. Effettivamente, quando il ricevente riceve $c(i)$, lo decifra con K_s per ottenere $s(i) = m(i) \oplus c(i-1)$, dato che il destinatario conosce anche $c(i-1)$, otterrà il blocco di testo in chiaro $m(i) = s(i) \oplus c(i-1)$. In secondo luogo, anche se due blocchi di testo in chiaro sono uguali, i corrispondenti cifrati saranno, quasi sempre, diversi. Infine, nonostante il mittente invii IV in chiaro, aumenta la richiesta di banda in caso di messaggi lunghi centinaia di blocchi solo di poco.

Consideriamo come esempio il blocco cifrato a 3 bit della Tabella 8.1 con testo in chiaro 010010010 e IV = $c(0) = 001$. Il mittente prima usa IV per calcolare $c(1) = K_s\,(m(1) \oplus c(0)) = 100$. Il mittente quindi calcola $c(2) = K_S(m(2) \oplus c(1)) = K_S(010 \oplus 100) = 000$, e $c(3) = K_S(m(3) \oplus c(2)) = K_S(010 \oplus 000) = 101$. Il lettore verifichi che il ricevente, conoscendo IV e K_S possa ricostruire il testo originale.

CBC ha un'importante ricaduta sulla progettazione dei protocolli di rete sicuri: è necessario fornire un meccanismo all'interno del protocollo per distribuire le stringhe IV del mittente e del ricevente. Vedremo come ciò avviene per molti protocolli più avanti in questo capitolo.

8.2.2 Crittografia a chiave pubblica

Per oltre 2000 anni (dai tempi di Cesare agli anni '70), la comunicazione cifrata richiedeva che gli interlocutori condividessero la conoscenza di un segreto comune: la chiave simmetrica usata per la cifratura e decifratura. Ciò comportava, però, che le due parti dovessero in qualche modo accordarsi sulla chiave condivisa tramite un canale presumibilmente sicuro. Forse le parti potevano prima incontrarsi di persona e accordarsi sulla chiave: due centurioni potevano darsi appuntamento alle terme di Roma e scambiarsi il cifrario, per poi cominciare la cifratura. Nell'ambito delle reti, le parti potrebbero non avere mai la possibilità di incontrasi di persona e conversare, se non sulla rete stessa. Ma è possibile inviare messaggi cifrati senza essersi precedentemente accordati su una chiave segretamente condivisa? La soluzione a questo problema fu proposta nel 1976 da Diffie e Hellman [Diffie 1976]: si trattava di un algoritmo (ora conosciuto come **scambio di chiave di Diffie-Hellman**) che ha portato allo sviluppo degli attuali sistemi di crittografia a chiave pubblica, particolarmente utili anche per l'autenticazione e le firme digitali. È interessante notare che si è di recente venuti a conoscenza del fatto che idee simili a quelle esposte in [Diffie 1976] e [RSA 1978] erano state sviluppate nei primi anni '70 in una serie di rapporti riservati del Communications-Electronics Security Group, in Gran Bretagna [Ellis 1987].

L'utilizzo della **crittografia a chiave pubblica** è concettualmente molto semplice. Supponiamo che Alice voglia comunicare con Bob. Come mostra la Figura 8.6, quest'ultimo (il destinatario dei messaggi) non possiede un'unica chiave segreta (come nel caso dei sistemi a chiave simmetrica), ma due: una **pubblica** (disponibile a chiunque, compresa Trudy) e una **privata**, che soltanto lui conosce. Useremo le notazioni K_B^+ e K_B^- per indicare rispettivamente la chiave pubblica e quella privata di Bob. Per comunicare con Bob, Alice prima di tutto si procura la chiave pubblica di Bob e codifica quindi il suo testo in chiaro (m), utilizzando la chiave pubblica di Bob e un dato (per esempio standardizzato) algoritmo di cifratura, e genera un messaggio criptato che indicheremo con $K_B^+(m)$. Quando Bob riceve il messaggio cifrato, utilizza la sua

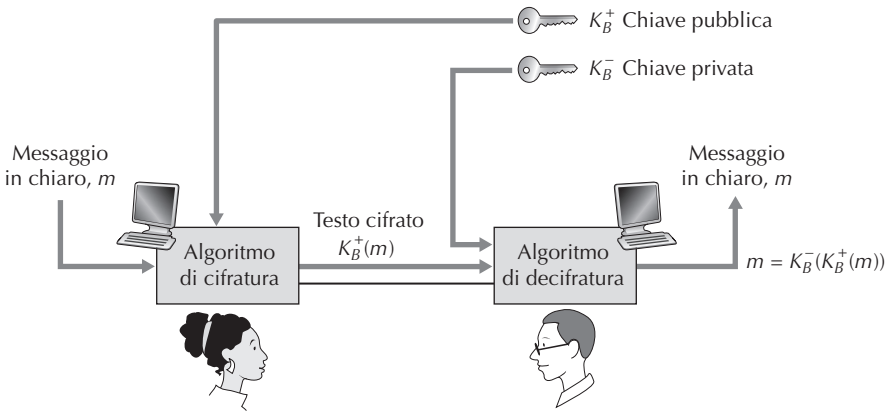

Figura 8.6 Crittografia a chiave pubblica.

chiave privata e un algoritmo per decodificarlo. In altri termini, Bob calcola $K_B^-(K_B^+(m))$. Vedremo in seguito che esistono algoritmi e tecniche per scegliere chiavi pubbliche e private in modo che $K_B^-(K_B^+(m)) = m$; cioè, applicando la chiave pubblica di Bob, K_B^+ a un messaggio m, e quindi applicando la chiave privata di Bob, K_B^- alla versione cifrata di m; cioè calcolando $K_B^-(K_B^+(m))$ si ottiene nuovamente m. Alice può utilizzare la chiave pubblica di Bob per inviargli un messaggio segreto senza che nessuno di loro debba distribuire chiavi segrete. Vedremo tra poco che invertendo la cifratura con chiave pubblica e quella con chiave privata si ottiene lo stesso risultato, cioè, $K_B^-(K_B^+(m)) = K_B^-(K_B^+(m)) = m$.

L'impiego della crittografia a chiave pubblica è quindi concettualmente semplice. Ma vengono subito in mente due obiezioni. La prima è che l'intruso che intercetta il messaggio potrà venire a conoscenza sia della chiave pubblica del destinatario sia dell'algoritmo che il mittente ha usato per la codifica. Nel nostro caso Trudy può tentare un attacco con testo in chiaro scelto, impiegando l'algoritmo di cifratura standard e la chiave pubblica di Bob per codificare i messaggi. Trudy può cercare, per esempio, di codificare messaggi, o parti di messaggio, che suppone potrebbero essere stati inviati da Alice. Chiaramente, affinché la crittografia a chiave pubblica funzioni, sia la chiave sia la codifica e decodifica devono essere scelte in maniera tale da rendere praticamente impossibile (o almeno così difficile da essere praticamente impossibile) all'intruso di determinare la chiave privata di Bob o decifrare in altri modi il messaggio che Alice gli ha inviato. Inoltre, dato che la chiave di Bob è pubblica, chiunque potrebbe inviargli un messaggio cifrato, magari fingendo di essere Alice. Con una singola chiave condivisa, il solo fatto che il trasmittente conosca il codice segreto costituisce un'implicita convalida della sua identità per il ricevente. Con la chiave pubblica questo non si verifica, in quanto chiunque può inviare un messaggio cifrato a Bob utilizzando la sua chiave che, come dice il nome, è pubblicamente disponibile. Per associare un trasmittente a un messaggio è quindi necessaria la firma digitale (Paragrafo 8.3).

RSA

Sebbene esistano numerosi algoritmi che soddisfano i requisiti esposti, l'**algoritmo RSA** (acronimo derivato dal nome dei suoi autori: Ron Rivest, Adi Shamir e Leonard Adleman) è diventato praticamente sinonimo di crittografia a chiave pubblica. Vediamo innanzitutto come e perché funziona.

RSA fa largo uso delle operazioni aritmetiche in modulo n. Vediamone ora brevemente l'aritmetica. Ricordiamo che $x \bmod n$ è il resto della divisione tra x e n; quindi, per esempio, $19 \bmod 5 = 4$. Nell'aritmetica in modulo si possono effettuare le usuali operazioni di addizione, moltiplicazione e di elevamento a potenza. Tuttavia, in tale aritmetica, il risultato di ognuna di queste operazioni è il resto intero ottenuto dalla divisione per n del risultato. Le operazioni di addizione e moltiplicazione sono facilitate dalle seguenti formule:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

$$[(a \bmod n) \bullet (b \bmod n)] \bmod n = (a \bullet b) \bmod n$$

Un'altra identità molto utile che può essere facilmente derivata dalla terza delle uguaglianze precedenti è: $(a \bmod n)^b \bmod n = a^b \bmod n$.

Supponiamo ora che Alice voglia inviare a Bob un messaggio cifrato con RSA, come mostrato nella Figura 8.6. Ricordiamo che qualsiasi messaggio non è altro che una sequenza di bit che può essere rappresentata in modo univoco da un numero intero insieme alla sua lunghezza. Per esempio, supponiamo che il messaggio sia la sequenza di bit 1001; tale messaggio può essere rappresentato dal numero intero in rappresentazione decimale 9. Quindi, cifrare un messaggio con RSA, è equivalente a cifrare un unico numero intero che lo rappresenta.

Due sono i punti principali su cui si basa RSA:

- la scelta della chiave pubblica e di quella privata;
- gli algoritmi di cifratura e di decifratura.

Per generare la chiave pubblica e quella privata, Bob deve eseguire i seguenti passi.

1. Scegliere due numeri primi, p e q: tanto più grande sarà il loro valore tanto più difficile risulterà violare RSA anche se, ovviamente, cifratura e decifratura richiederanno più tempo. Gli RSA Laboratories raccomandano che il prodotto di p e q sia dell'ordine di 1024 bit. Per approfondimenti su come individuare numeri primi grandi, si veda [Caldwell 2012].
2. Calcolare $n = pq$ e $z = (p - 1)(q - 1)$.
3. Scegliere un numero e (*encryption*) minore di n, diverso da 1 e relativamente primo a z (ovvero che non abbia divisori in comune con z).
4. Trovare un numero d (*decryption*) tale che $ed - 1$ sia divisibile (cioè non vi sia resto) da z. In altri termini, dato e, d è scelto in modo tale che

   $$ed \bmod z = 1$$

5. La chiave pubblica di Bob, K_B^+, è la sequenza dei bit concatenati della coppia (n, e); quella privata, K_B^-, è la coppia (n, d).

Nel nostro esempio, la cifratura di Alice e la decifratura di Bob sono eseguite come segue.

- Supponiamo che Alice voglia inviare a Bob una stringa di bit, o un numero $m < n$. Per la codifica, Alice calcola m^e e poi il resto intero di m^e/n. Allora, il messaggio cifrato c inviato risulta:

  $$c = m^e \bmod n$$

  La sequenza di bit corrispondente al testo cifrato c viene inviata a Bob.

- Per decifrare il messaggio ricevuto, Bob calcola

  $$m = c^d \bmod n$$

  che richiede l'utilizzo della sua chiave segreta (n, d).

Tabella 8.2 Codifica RSA di Alice, $e = 5$, $n = 35$.

Lettere in chiaro	m: rappresentazione numerica	m^e	Testo cifrato $c = m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Un semplice esempio di RSA vede Bob scegliere: $p = 5$ e $q = 7$ (due valori troppo piccoli per essere sicuri); $n = 35$ e $z = 24$; $e = 5$, poiché 5 e 24 non hanno divisori comuni; $d = 29$, perché $5 \times 29 - 1$ (cioè, $ed - 1$) è divisibile per 24. Quindi, rende pubblica la coppia (n, e), cioè (35, 5), e mantiene segreto il valore $d = 29$. Supponiamo ora che Alice, che è a conoscenza di questi due valori, voglia inviare le lettere "l" "o" "v" ed "e" a Bob. Interpretando le lettere come numeri fra 1 e 26, Alice e Bob procedono alla codifica e alla decodifica come mostrato nelle Tabelle 8.2 e 8.3. Si noti che nell'esempio ognuna delle quattro lettere è considerata un messaggio distinto. Un esempio più realistico sarebbe la conversione delle quattro lettere in una rappresentazione ASCII a 8 bit e quindi la cifratura dell'intero corrispondente alla sequenza di 32 bit, ma tale esempio realistico genererebbe numeri troppo lunghi da scrivere in un libro.

Dato che l'esempio nelle Tabelle 8.2 e 8.3 ha già prodotto numeri estremamente grandi e dato che sappiamo che p e q dovrebbero essere ciascuno lungo molte centinaia di bit, sorgono alcune questioni pratiche relative a RSA. Per approfondire questi importanti aspetti pratici che riguardano il sistema RSA, i numeri primi molto grandi, la scelta di e e d e come calcolare la potenza di numeri grandi, temi che esulano dalle finalità di questo testo, vi rimandiamo a [Kaufman 1995].

Chiavi di sessione

Notiamo che l'elevamento a potenza richiesto da RSA è un processo che impiega molto tempo. Per contro, DES è almeno 100 volte più veloce di RSA, e a livello hardware il divario è ancora maggiore, essendo 1000 o 10.000 volte più veloce [RSA

Tabella 8.3 Decodifica RSA di Bob, $d = 29$, $n = 35$.

Testo cifrato c	c^d	$m = c^d \bmod n$	Lettere in chiaro
17	481968572106750915091411825223071697	12	l
15	12783403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	10000000000000000000000000000000	5	e

Fast 2012]. Come risultato, nella prassi comune, RSA è sovente utilizzato congiuntamente alla crittografia a chiave simmetrica. Così, se Alice vuole inviare a Bob un gran numero di dati cifrati, per codificarli può innanzitutto scegliere la chiave che sarà usata per codificare i dati stessi, detta **chiave di sessione** K_S. Alice deve informare Bob della chiave di sessione, poiché è la chiave simmetrica condivisa che sarà usata nel cifrario a chiave simmetrica (per esempio DES o AES). Quindi, ne codifica il valore con la chiave pubblica RSA di Bob: cioè, calcola $c = (K_S)^e \bmod n$. Quando questi riceve la chiave di sessione cifrata con RSA, c, la decifra e ottiene la chiave di sessione K_S, utilizzata da Alice per il trasferimento di dati cifrati.

Perché RSA funziona?

Ma come avviene esattamente la cifratura e la decifratura con RSA? Per rispondere a questa domanda dobbiamo avvalerci dell'aritmetica in modulo, per cui il risultato delle operazioni di addizione, moltiplicazione ed elevamento a potenza è sostituito dal resto intero della divisione per n del risultato. Poniamo $n = pq$ dove p e q sono i numeri primi utilizzati nell'algoritmo RSA.

Ricordiamo che nella codifica RSA, si calcola

$$c = m^e \bmod n$$

e che nella decodifica si eleva questo valore alla potenza d, ancora in aritmetica modulo n. Il risultato delle due fasi è quindi $(m^e \bmod n)^d \bmod n$. Ricordiamo un'importante proprietà dell'aritmetica in modulo: $(a \bmod n)^d \bmod n = a^d \bmod n$ per ogni a, n e d.

Abbiamo quindi, usando $a = m^e$ nella proprietà:

$$(m^e \bmod n)^d = m^{ed} \bmod n$$

A questo punto rimane da dimostrare che $m^{ed} \bmod n = m$. Per farlo dobbiamo ricorrere a un risultato "magico" che proviene dalla teoria dei numeri. In particolare, facciamo riferimento al seguente teorema: se p e q sono primi, $n = pq$ e $z = (p - 1)(q - 1)$, allora $x^y \bmod n = x^{(y \bmod z)} \bmod n$ [Kaufman 1995]. Applicando questo risultato possiamo scrivere

$$(m^e)^d \bmod n = m^{(ed \bmod (pz))} \bmod n$$

Ma ricordiamo che abbiamo scelto e e d tali che $ed \bmod z = 1$, abbiamo

$$(m^e)^d \bmod n = m^1 \bmod n = m$$

Questo è il risultato che volevamo. Con il primo elevamento a potenza di e (cifratura) seguito dall'elevamento a potenza di d (decifratura), otteniamo il valore originale, m. Ancor più notevole è il fatto che se invertiamo l'ordine degli elevamenti a potenza, eseguendo prima la decifratura e poi la cifratura, otteniamo nuovamente il valore originale, m. La dimostrazione di questo fatto è basata sulla semplice osservazione che:

$$(m^d \bmod n)^e \bmod n = (m^d)^e \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

L'efficacia di RSA consiste nel fatto che non si conoscono algoritmi veloci per la fattorizzazione dei numeri interi. Quindi, anche conoscendo il valore pubblico n risulta computazionalmente proibitivo determinare i fattori primi p e q (con i quali, dato il

valore pubblico *e,* si può facilmente calcolare la chiave segreta). D'altra parte, non si sa se esista o no un algoritmo veloce per la fattorizzazione di un numero, ed è in questo senso che la sicurezza di RSA non è garantita.

Un altro diffuso algoritmo di cifratura è quello di Diffie-Hellman, discusso in uno dei problemi di fine capitolo. È meno versatile di RSA in quanto non può cifrare messaggi di lunghezza arbitraria, ma può stabilire una chiave simmetrica usata nel cifrare i messaggi.

8.3 Integrità dei messaggi e firma digitale

Nel paragrafo precedente abbiamo visto come la cifratura può essere usata per garantire riservatezza tra due entità comunicanti. In questo paragrafo rivolgiamo l'attenzione all'altro tema egualmente importante che riguarda l'uso della crittografia per garantire l'**integrità dei messaggi**, tema noto anche come **autenticazione dei messaggi**. Vedremo che le funzioni hash crittografiche sono componenti comuni sia dell'integrità dei messaggi sia della firma digitale e dell'autenticazione dei partecipanti.

Definiamo il problema dell'integrità dei messaggi usando, ancora una volta, Alice e Bob. Supponete che Bob riceva un messaggio, che può essere cifrato o in chiaro, e che creda che sia stato inviato da Alice. Per autenticare questo messaggio, Bob deve verificare:

1. che il messaggio sia stato effettivamente originato da Alice;
2. che non sia stato alterato lungo il percorso verso Bob.

Vedremo nei Paragrafi 8.4 – 8.7 che il problema dell'integrità dei messaggi è un aspetto critico in tutti i protocolli di rete sicuri.

Come esempio specifico considerate una rete di calcolatori che usa un algoritmo di instradamento di tipo link state, come OSPF, per determinare le rotte tra ciascuna coppia di router nella rete (si veda il Capitolo 5). In un algoritmo link state, ciascun router ha necessità di inviare in broadcast un messaggio sullo stato dei suoi collegamenti a tutti gli altri router della rete. Questo messaggio contiene una lista di tutti i router vicini e direttamente connessi, nonché i costi per raggiungerli. Una volta che il router ha ricevuto i messaggi di stato da tutti gli altri router, può creare una mappa completa della rete, eseguire il proprio algoritmo di instradamento a costo minimo e configurare le proprie tabelle di inoltro. Un attacco relativamente semplice all'algoritmo di instradamento è quello di distribuire messaggi fasulli con informazioni errate. Ne consegue la necessità dell'integrità dei messaggi: quando il router B riceve un messaggio da un router A, il router B dovrebbe verificare che è stato proprio A a creare il messaggio e, inoltre, che nessuno ha alterato il messaggio in transito.

In questo paragrafo descriveremo le tecniche comunemente usate per garantire l'integrità dei messaggi adottate da molti protocolli di rete sicuri. Ma, prima di farlo, dobbiamo trattare un altro argomento importante della crittografia: le funzioni hash crittografiche.

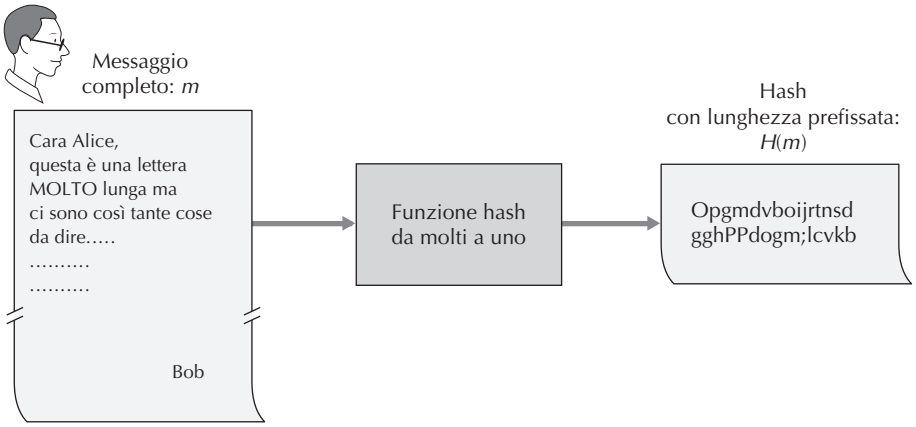

Figura 8.7 Funzioni di hash.

8.3.1 Funzioni hash crittografiche

Una **funzione hash** prende un ingresso, *m*, e calcola una stringa di lunghezza fissata, detta hash (Figura 8.7). I checksum Internet (Capitolo 3) e CRC (Capitolo 6) soddisfano questa definizione. Una **funzione hash crittografica** deve però soddisfare un'ulteriore proprietà:

- deve essere computazionalmente impossibile trovare due messaggi x e y diversi, tali che $H(x) = H(y)$.

Vale a dire che, dal punto di vista computazionale, un malintenzionato non deve avere alcuna possibilità di poter sostituire un messaggio con un altro messaggio che sia protetto dalla funzione hash, cioè data la coppia messaggio-hash (m, $H(m)$), creata dal trasmittente, un intruso non può falsificare il contenuto di un altro messaggio, y, che abbia lo stesso valore hash dell'originale.

Il checksum semplice, come quello di Internet, è un algoritmo di hash crittografico poco efficace. In questo caso, anziché usare l'aritmetica con complemento a 1, trattiamo i caratteri come byte sommandoli a blocchi di 4. Supponiamo che Bob debba ad Alice 100 dollari e 99 centesimi e che come attestato di tale debito le invii la stringa di testo "IOU100.99BOB".[2] La rappresentazione ASCII (in notazione esadecimale) di questi caratteri è 49, 4F, 55, 31, 30, 30, 2E, 39, 39, 42, 4F, 42.

Il checksum a 4 byte di questo messaggio è B2 C1 D2 AC (Figura 8.8, in alto). Ma proprio qui sorge il problema: la stringa "I0U900.19BOB", leggermente differente dalla prima (ma molto più onerosa per Bob) presenta lo *stesso* checksum della precedente. Di conseguenza, questo algoritmo viola i requisiti di autenticazione pre-

[2] Leggendo l'una dopo l'altra, con pronuncia inglese, le lettere "IOU" otteniamo la frase "I owe you" che significa: "Io ti devo" (*N.d.R.*).

Messaggio	Rappresentazione ASCII				
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	Checksum

Messaggio	Rappresentazione ASCII				
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	Checksum

Figura 8.8 Il messaggio originale e il messaggio fraudolento hanno lo stesso checksum.

cedentemente indicati, in quanto risulta relativamente semplice trovare altri dati con il medesimo checksum del messaggio originale. È ovvio che i requisiti di sicurezza richiedono l'impiego di una funzione hash più efficiente.

Un algoritmo molto utilizzato per l'hash dei messaggi è MD5, ideato da Ron Rivest [RFC 1321], in grado di calcolare una hash di 128 bit con un processo a quattro fasi. Si inizia con la normalizzazione attraverso l'aggiunta del valore 1 seguito da una serie di 0, in un numero tale da soddisfare determinate condizioni. Le successive fasi prevedono: l'aggiunta in coda di una rappresentazione a 64 bit della lunghezza del messaggio originale, l'inizializzazione di un accumulatore e, per ultimo, un passaggio in cui i blocchi composti da 16 gruppi di 32 bit (word) del messaggio vengono "triturati" attraverso un processo che prevede quattro cicli di elaborazione. Per una descrizione dettagliata di MD5 (inclusa una sua implementazione in linguaggio C) si veda [RFC 1321].

Un altro importante algoritmo hash, attualmente in uso, è **hash sicuro** (SHA-1, *secure hash algorithm*) descritto in [FIPS 1995], basato su princìpi simili a quelli utilizzati nel progetto di una versione precedente dell'algoritmo di Rivest, MD4 [RFC 1320]. SHA-1 è uno standard utilizzato nelle applicazioni federali USA in cui è richiesto un algoritmo hash crittografico e produce una sintesi del messaggio di 160 bit. La maggior lunghezza del risultato rende SHA-1 più sicuro.

8.3.2 Codice di autenticazione dei messaggi

Ritorniamo al problema dell'integrità dei messaggi. Ora che abbiamo compreso le funzioni hash, facciamo un primo tentativo di come dovremmo realizzare l'integrità di un messaggio.

1. Alice crea un messaggio m e calcola la stringa hash $h = H(m)$, per esempio con SHA-1.

2. Alice aggiunge h al messaggio m, creando il messaggio esteso (m, h) e lo manda a Bob.
3. Bob riceve il messaggio esteso (m, h) e calcola $H(m)$. Se $H(m) = h$, Bob conclude che va tutto bene.

Questo approccio presenta ovviamente un difetto: Trudy può creare un messaggio falso, m', nel quale dice di essere Alice, calcola e manda a Bob $(m', H(m'))$. Quando Bob riceve il messaggio, tutto va a buon fine nel passo 3 e Bob non sospetta alcuna attività insolita.

Per garantire l'integrità dei messaggi, oltre alle funzioni hash crittografiche, Alice e Bob hanno bisogno di un segreto condiviso. Questo segreto condiviso, che non è altro che una stringa di bit, è chiamato **chiave di autenticazione**. Usando questo segreto condiviso, l'integrità del messaggio è realizzata come segue.

1. Alice crea un messaggio m, concatena s con m per creare $m + s$, calcola la stringa hash $H(m + s)$, per esempio con SHA-1. $H(m + s)$ è chiamato **codice di autenticazione del messaggio** (MAC, *message authentication code*).
2. Alice aggiunge il *MAC* al messaggio m, creando il messaggio esteso $(m, H(m + s))$ e lo manda a Bob.
3. Bob riceve il messaggio esteso (m, h) e, avendo ricevuto m e conoscendo s, calcola il MAC $H(m + s)$. Se $H(m + s) = h$, Bob conclude che va tutto bene.

Un riassunto della procedura è illustrato nella Figura 8.9. Si noti che il MAC, che qui sta per *Message Authentication Code*, non è lo stesso MAC usato nei protocolli a livello di collegamento, che in quel caso sta per *Medium Access Control*.

Una interessante caratteristica di MAC è che non richiede un algoritmo di cifratura. Effettivamente, in molte applicazioni, compresi gli algoritmi link state descritti precedentemente, le entità in comunicazione si devono preoccupare solo dell'integrità del messaggio e non della sua riservatezza. Usando MAC, le entità possono autenti-

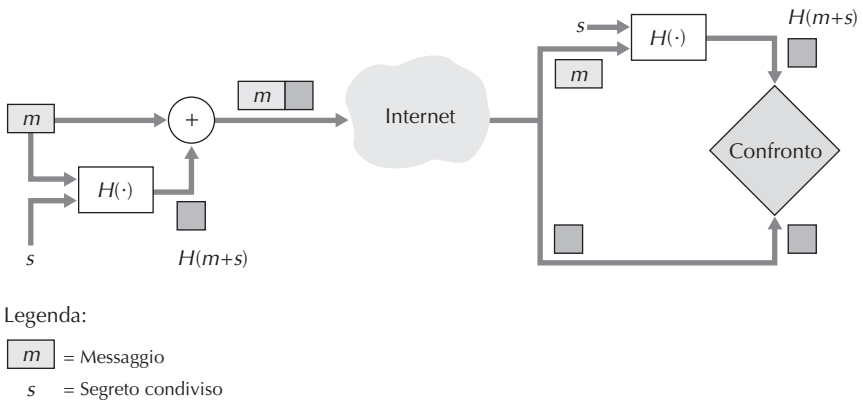

Figura 8.9 Codice di autenticazione del messaggio (MAC).

care i messaggi che si scambiano, senza che debbano includere complessi algoritmi di cifratura nel processo per garantire integrità del messaggio.

Negli anni sono stati proposti parecchi standard diversi per MAC. Il più comune oggi è **HMAC**, che può essere usato sia con MD5 che con SHA-1. **HMAC**, in realtà, applica la funzione hash due volte ai dati e alla chiave di autenticazione [Kaufman 1995; RFC 2104].

Rimane ancora la questione importante di come venga distribuita la chiave di autenticazione condivisa tra le entità in comunicazione. Per esempio, nell'algoritmo di instradamento link state avremo bisogno di distribuire in un qualche modo la chiave di autenticazione a ciascun router fidato nella rete di calcolatori. Si noti che tutti i router possono usare la stessa chiave di autenticazione. Un amministratore di rete può, a dire il vero, svolgere questo compito manualmente, visitando ciascun router, oppure, se ciascun router ha una sua chiave pubblica, l'amministratore di rete può comunicare la chiave di autenticazione a ciascun router, cifrandola con la chiave pubblica del router e poi inviandogliela tramite la rete.

8.3.3 Firme digitali

Pensate a quante firme avete apposto su assegni, lettere e atti attestando in questo modo che ne approvate il contenuto. Nel mondo digitale, per indicare il titolare o il creatore di un documento, o dichiarare di essere d'accordo con il suo contenuto, si ricorre alla **firma digitale**, una tecnica di crittografia che consente di raggiungere svariati obiettivi.

Come la firma tradizionale, anche quella digitale deve essere verificabile e non falsificabile. Cioè, deve consentire di dimostrare che un certo documento sia davvero stato firmato proprio da quella data persona (verificabile) e che solo lei poteva realizzarlo (non falsificabile).

Consideriamo ora come progettare uno schema di firma digitale. Si osservi che quando Bob firma un messaggio, deve inserire qualcosa nel messaggio che per lui è unico. Bob potrebbe considerare l'aggiunta di un MAC per la firma, dove il MAC è stato creato manipolando il messaggio con una funzione hash con una chiave segreta nota solo a Bob. Se Alice volesse verificare la firma, dovrebbe avere una copia della chiave e, in questo caso, la chiave non è unica per Bob. Il MAC, quindi, non è adatto a questo contesto.

Ricordiamo la cifratura a chiave pubblica: Bob ha una chiave pubblica e una privata, che per lui sono uniche. La cifratura a chiave pubblica, quindi, è un eccellente candidato alla firma digitale. Esaminiamo come funziona.

Supponiamo che Bob voglia firmare digitalmente un documento, *m.* Possiamo pensare il documento come un file o un messaggio che Bob deve firmare e inviare. Come illustrato nella Figura 8.10, per firmare il documento utilizza la sua chiave privata, K_B^-, per calcolare $K_B^-(m)$. Può sembrare strano che impieghi questa tecnica (che nel Paragrafo 8.2 abbiamo visto essere usata nella decodifica di un messaggio cifrato con la chiave pubblica) per firmare un documento. Ma ricordiamo che cifratura e decifratura non sono altro che operazioni matematiche (elevazione a esponente della

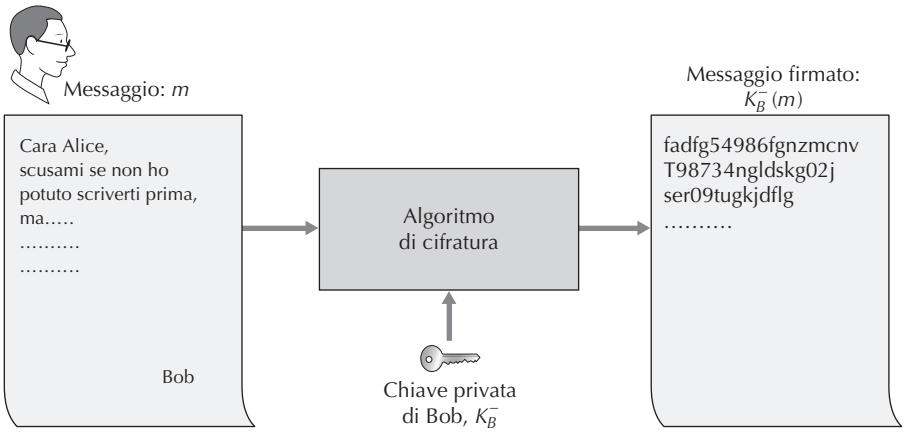

Figura 8.10 Creazione della firma digitale di un documento.

potenza di e e d in RSA, nel Paragrafo 8.2) e che l'obiettivo di Bob non è confondere o oscurare il contenuto, ma firmare il documento in modo che sia verificabile e non falsificabile.

Vediamo, ora, se la firma digitale $K_B^-(m)$ soddisfa questi requisiti. Supponiamo che Alice ottenga m e $K_B^-(m)$ e voglia convincere un giudice che il documento è stato effettivamente firmato da Bob e che lui era la sola persona che poteva farlo. Alice applica la chiave pubblica di Bob e calcola $K_B^+(K_B^-(m))$ e riproduce m, che corrisponde esattamente al documento originale. Il giudice deduce che solo Bob può aver apposto la firma, per i seguenti motivi.

- Chiunque abbia firmato il messaggio deve aver utilizzato la chiave di cifratura privata, K_B^-, nel calcolo della firma $K_B^-(m)$, in modo che $K_B^+(K_B^-(m)) = m$.
- Presupponendo che Bob non abbia dato a nessuno la propria chiave e che questa non sia stata rubata, allora solo lui poteva conoscere K_B^-. Infatti (Paragrafo 8.2) la conoscenza della chiave pubblica, K_B^+, non è d'aiuto nello scoprire la chiave privata, K_B^-.

È anche importante notare che la firma creata da Bob per m non sarà valida per nessun altro messaggio in quanto se m' è un qualunque altro messaggio, $K_B^+(K_B^-(m'))$ non sarà uguale a m'. Di conseguenza, le tecniche di crittografia a chiave pubblica forniscono anche l'integrità del messaggio, consentendo al ricevente di verificare che il messaggio era inalterato come alla sorgente.

L'utilizzo della crittografia a chiave pubblica per le firme digitali presenta il problema per cui cifratura e decifratura dei dati sono onerose dal punto di vista computazionale. Certo, in molti casi, la firma attraverso cifratura e decifratura completa dei dati potrebbe risultare eccessiva. Un approccio più efficiente è rappresentato dall'introduzione delle funzioni hash nella firma digitale. Ricordiamo (Paragrafo 8.3.2) che l'algoritmo prende il messaggio, m, di lunghezza arbitraria ed elabora una "impronta

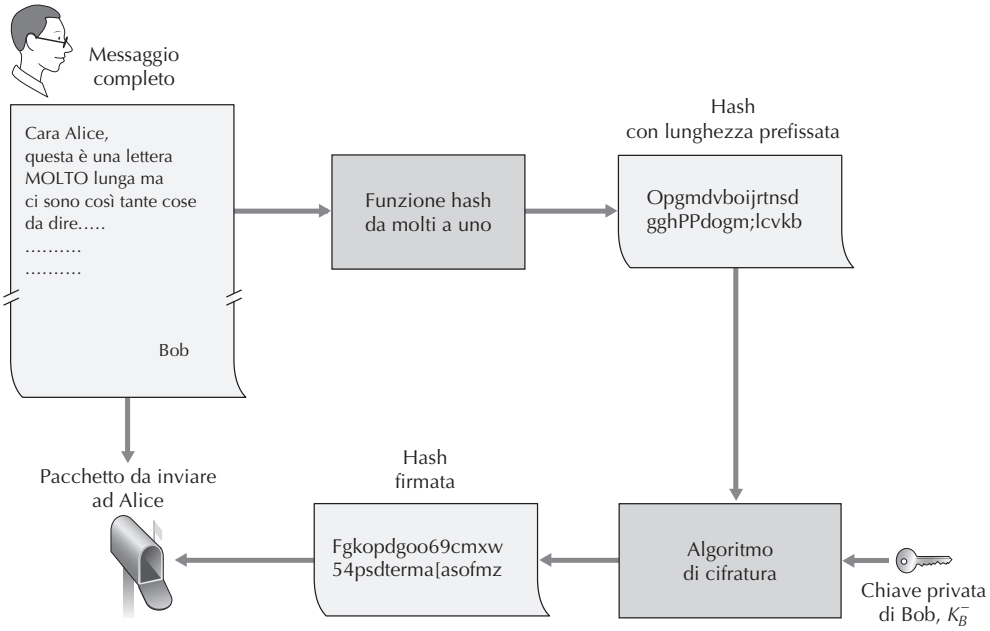

Figura 8.11 Invio di un messaggio firmato digitalmente.

digitale" dei dati di lunghezza prefissata $H(m)$. Bob invece di firmare l'intero messaggio con $K_B^-(m)$, potrebbe limitarsi a firmare solo l'hash, calcolando $K_B^-(H(m))$. Poiché $H(m)$ è generalmente più piccolo del messaggio originale m, lo sforzo computazionale per creare la firma digitale è sostanzialmente diminuito.

La Figura 8.11 fornisce un riassunto della procedura per creare la firma digitale. Nel nostro esempio, il processo di firma digitale vede Bob applicare una funzione hash all'intero messaggio e ottenerne una hash, che cifra con la propria chiave privata. L'originale (con testo in chiaro) e la sua sintesi firmata digitalmente (d'ora innanzi chiamata firma digitale) sono quindi inviati ad Alice. Il processo di verifica dell'integrità del messaggio vede Alice per prima cosa applicare la chiave pubblica del trasmittente al messaggio per ottenerne una hash e poi impiegare la funzione hash al messaggio con il testo in chiaro, in modo da procurarsi la seconda hash. Se i due risultati coincidono, Alice può essere tranquilla dell'integrità del messaggio e sull'identità del suo autore (Figura 8.12).

Prima di continuare, confrontiamo brevemente firma digitale e MAC, in quanto presentano alcune corrispondenze, ma anche delle differenze sottili e importanti. Entrambi partono da un messaggio o da un documento. Per creare un MAC dal messaggio, gli aggiungiamo la chiave di autenticazione e prendiamo l'hash del risultato. Si noti che non sono coinvolte né la crittografia a chiave simmetrica né quella a chiave pubblica. Per creare la firma digitale, prima prendiamo l'hash del messaggio e poi la cifriamo con la nostra chiave privata (usando la crittografia a chiave pubblica). Una firma digitale, quindi, è una tecnica più gravosa, in quanto richiede l'infrastruttura di

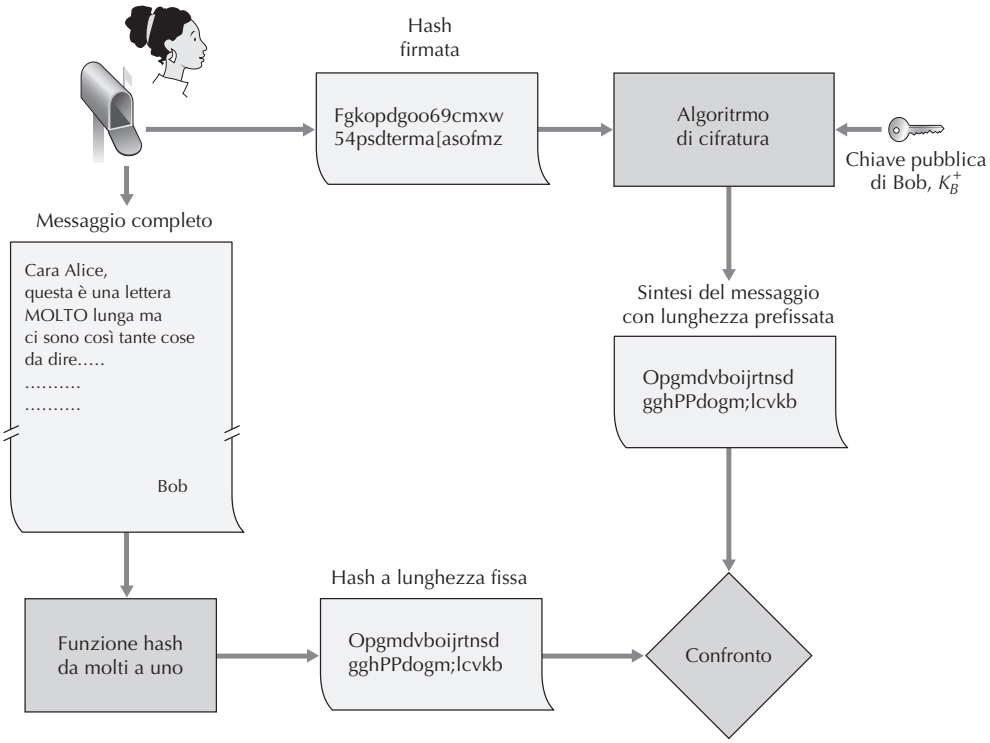

Figura 8.12 Controllo dell'integrità del messaggio firmato.

chiave pubblica (PKI, *public key infrastructure*) sottostante, con le relative autorità di certificazione (descritte in seguito). Vedremo nel Paragrafo 8.4 che PGP, un comune sistema di sicurezza per la posta elettronica, usa la firma digitale per garantire l'integrità dei messaggi. Abbiamo già visto che OSPF usa MAC per l'integrità dei messaggi e vedremo che MAC viene usato anche da altri protocolli di sicurezza a livello di trasporto e di rete nei Paragrafi 8.5 e 8.6.

Certificazione della chiave pubblica

Un'importante applicazione della firma digitale è la **certificazione della chiave pubblica**, cioè la certificazione che una chiave pubblica appartenga a una specifica entità. La certificazione della chiave pubblica è usata in molti protocolli per la sicurezza di rete, compresi IPsec e SSL.

Per analizzare questo problema, consideriamo un esempio di e-commerce. Supponiamo che Alice svolga un'attività di consegna pizze a domicilio e che accetti ordini attraverso Internet. Bob, che adora la pizza, le invia un messaggio, con testo in chiaro, in cui inserisce il suo indirizzo e il tipo di pizza desiderato. Inserisce anche la firma digitale, cioè un hash firmato del messaggio originale in chiaro, per provare ad Alice che è lui la vera sorgente del messaggio. Alice può ottenere la chiave pubblica di Bob, magari da un server di chiavi pubbliche o dal messaggio di posta e verificare

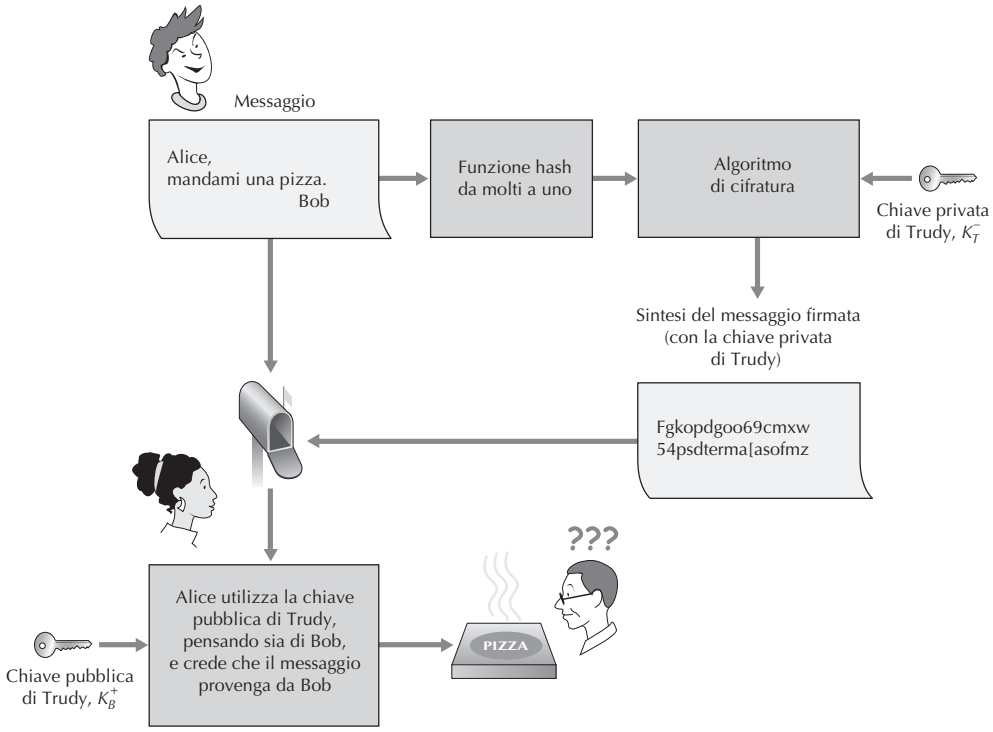

Figura 8.13 Trudy finge di essere Bob utilizzando la crittografia a chiave pubblica.

la firma digitale e accertarsi, in questo modo, che sia stato proprio lui, e non qualche "burlone", a fare l'ordine.

Tutto ciò funziona bene fino a quando l'abile Trudy (Figura 8.13) decide di fare uno scherzo. Fingendo di essere Bob invia la richiesta per la consegna di una pizza con l'indirizzo di quest'ultimo. Appone anche la firma digitale, ma per firmare la hash del messaggio utilizza la propria chiave privata. Inoltre, allega la propria chiave pubblica dicendo che è quella di Bob. Di conseguenza, Alice applicherà la chiave pubblica di Trudy (pensando che sia quella di Bob) alla firma digitale e concluderà che il messaggio in chiaro è stato creato da Bob che resterà, invece, molto sorpreso quando il fattorino gli consegnerà la pizza che non ha ordinato.

Questo esempio mostra che, per utilizzare la crittografia a chiave pubblica, utenti, browser, router e così via devono avere la certezza che la chiave pubblica sia proprio quella del corrispondente. Per esempio, quando Alice vuole comunicare con Bob usando la crittografia a chiave pubblica, deve verificare che la chiave pubblica, che suppone essere di Bob, sia effettivamente di quest'ultimo.

Generalmente, la relazione tra una data chiave pubblica e una determinata entità è stabilita da un'**autorità di certificazione** (CA, *certification authority*), il cui compito è di validare l'identità ed emettere certificati e ha il seguenti ruoli.

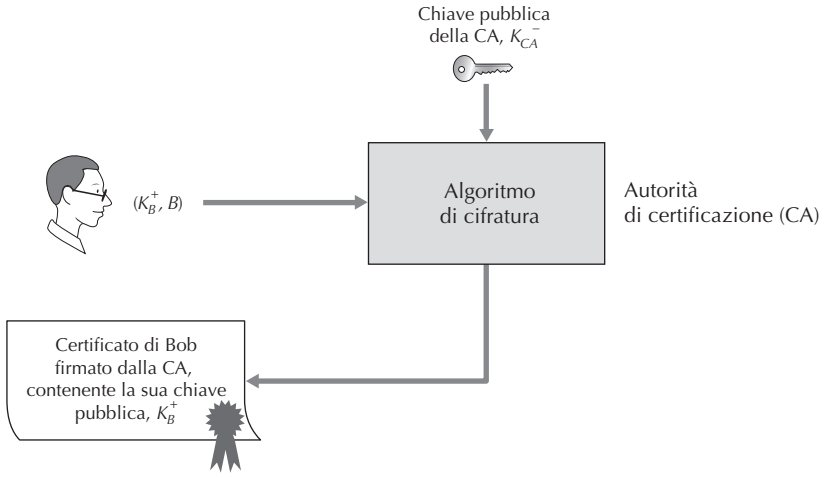

Figura 8.14 Bob ottiene un certificato da una CA.

1. Verifica che un'entità (persona fisica, router o altro) sia veramente chi afferma di essere. In effetti, non esistono specifiche procedure che stabiliscono come questa mansione debba essere svolta per cui, quando si definisce un accordo con una CA, occorre sperare che questa esegua un appropriato e rigoroso controllo dell'identità. Per esempio, se Trudy si presentasse all'autorità di certificazione di Fly-By-Night affermando "Sono Alice" e ne ricevesse la relativa certificazione, allora non si dovrebbe fare affidamento sulle chiavi pubbliche autenticate da quell'autorità. In conclusione, ci si può fidare dell'identità associata a una chiave pubblica solo se si è sicuri della CA e delle sue tecniche di accertamento.
2. Una volta verificata l'identità, la CA rilascia un **certificato** che autentica la corrispondenza fra chiave pubblica ed entità. Il certificato contiene la chiave pubblica e le informazioni di identificazione globali e uniche del suo proprietario (per esempio, il nome di una persona o un indirizzo IP), ed è firmato digitalmente dalla CA (Figura 8.14).

Vediamo ora come i certificati possono essere impiegati per contrastare burloni e malintenzionati. Quando Alice riceve l'ordine di Bob, cerca il certificato di questo e utilizza la chiave pubblica della CA per verificare l'affidabilità di quella inserita nel messaggio.

ITU (*international telecommunication union*) e IETF (*Internet engineering task force*) hanno sviluppato standard per le autorità di certificazione. La raccomandazione ITU X.509 [ITU 2005a] specifica il servizio di autenticazione e la sintassi dei certificati. [RFC 1422], che descrive la gestione delle chiavi basata su CA per le e-mail su Internet, è compatibile con X.509 e ne espande i contenuti inserendo procedure e convenzioni per la gestione delle chiavi. Nella Tabella 8.4 sono descritti alcuni tra i più significativi campi dei certificati.

Tabella 8.4 Campi di una chiave pubblica X.509 e RFC 1422.

Nome campo	Descrizione
Versione	Numero di versione della specifica X.509
Numero seriale	Identificatore unico del certificato fornito dalla CA
Firma	Specifica l'algoritmo utilizzato dalla CA per firmare il certificato
Nome dell'emittente	Identificativo della CA che rilascia il certificato, in formato DN [RFC 4514]
Periodo di validità	Inizio e fine del periodo di validità del certificato
Nome del soggetto	Identificativo dell'entità la cui chiave pubblica è associata al certificato (in formato DN)
Chiave pubblica del soggetto	Chiave pubblica del soggetto e indicazioni dell'algoritmo da utilizzare

8.4 Autenticazione di un punto terminale

Con **autenticazione di un punto terminale** (o *end-point authentication*) si intende il processo attraverso il quale una entità prova la sua identità a un'altra entità su una rete di calcolatori, come nel caso di un utente che prova la propria identità a un server e-mail, così come riconosciamo il viso di un amico quando lo incontriamo, la sua voce quando ci telefona, o ci identifichiamo a un pubblico ufficiale tramite la carta d'identità.

In questo paragrafo consideriamo come un'entità possa autenticarne un'altra mentre è in atto una comunicazione via rete. Ci concentreremo sull'autenticazione di un partecipante attivo durante la comunicazione. Esiste una sottile, ma sostanziale differenza rispetto al provare che un messaggio ricevuto è stato veramente inviato da un dato mittente: problema, quest'ultimo, che riguarda la **firma digitale** (Paragrafo 8.3).

Nell'autenticazione in rete, le parti comunicanti non possono scambiarsi informazioni biometriche, come l'aspetto del viso o il timbro della voce, in quanto le parti possono essere apparecchiature quali router o processi client e server. Quindi, l'autenticazione deve essere basata unicamente sullo scambio di messaggi o di dati come parte di un **protocollo di autenticazione**. Generalmente, questo dovrebbe intervenire *prima* che le due parti eseguano qualsiasi altro protocollo (per esempio, per il trasferimento dati o per lo scambio delle tabelle di instradamento o per un protocollo di posta elettronica). Soltanto dopo che è stata comprovata l'identità delle parti, queste possono iniziare a lavorare.

Come per `rdt` (Capitolo 3), anche in questo caso analizzeremo varie versioni di un protocollo di autenticazione, che chiameremo **ap** (*authentication protocol*), elencando pregi e difetti di ciascuna. In [Bryant 1988] potete trovare un'ipotetica corrispondenza tra i progettisti di un sistema di autenticazione in rete e la loro reazione alla scoperta dei molti sottili aspetti coinvolti. Cominciamo supponendo che Alice debba autenticarsi a Bob.

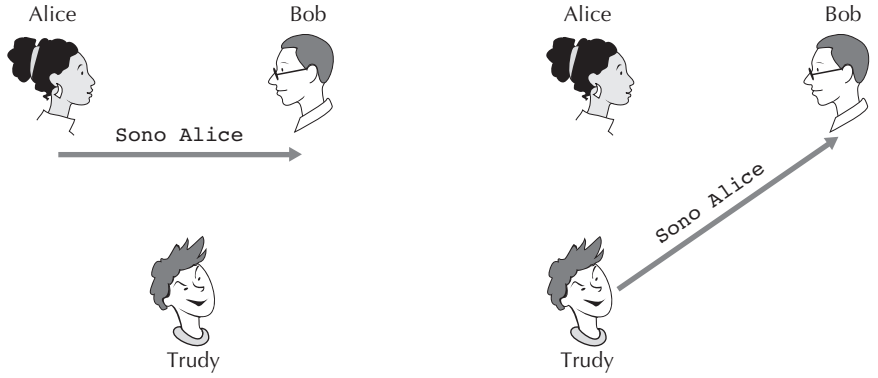

Figura 8.15 Scenario con fallimento del protocollo *ap1.0*.

8.4.1 Protocollo di autenticazione *ap1.0*

Forse il più semplice protocollo di autenticazione che possiamo immaginare è quello in cui Alice invia un messaggio a Bob dicendo semplicemente di essere Alice (Figura 8.15).

Ovviamente, il destinatario non può assolutamente essere sicuro che la persona che ha inviato il messaggio "Sono Alice" sia davvero Alice, perché anche Trudy potrebbe averlo fatto.

8.4.2 Protocollo di autenticazione *ap2.0*

Se Alice avesse un indirizzo di rete conosciuto (per esempio, l'indirizzo IP) che utilizza abitualmente, Bob potrebbe autenticarla verificando la corrispondenza fra l'indirizzo sorgente sul datagramma IP che trasporta il messaggio e quello di Alice. Questo forse dissuaderebbe un intruso molto ingenuo, ma non fermerebbe certo i lettori di questo libro, né molti altri.

Dal nostro studio in merito ai livelli di rete e di collegamento sappiamo che non è particolarmente difficile creare un datagramma IP contenente un indirizzo sorgente artefatto e poi inviarlo attraverso il protocollo a livello di collegamento al primo router. Ciò è possibile nel caso in cui si abbia accesso al codice del sistema operativo e si possa costruire un proprio kernel di sistema, come nel caso di Linux o di molti altri sistemi operativi di libera distribuzione. Da quel momento, il datagramma con il falso indirizzo sorgente sarà coscienziosamente instradato verso il destinatario. Questo approccio (Figura 8.16) è una forma di **spoofing** di IP. Lo spoofing (impersonificazione) potrebbe essere evitato se il router del primo hop del mittente malintenzionato fosse configurato in modo da inoltrare solo datagrammi con il suo indirizzo IP sorgente [RFC 2827]; funzionalità che non è universalmente adottata né imposta.

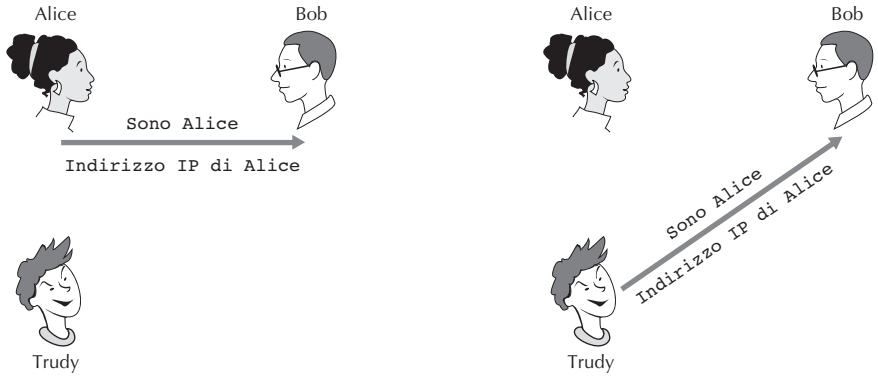

Figura 8.16 Scenario con fallimento del protocollo *ap2.0*.

8.4.3 Protocollo di autenticazione *ap3.0*

Un classico approccio all'autenticazione è l'utilizzo di una password. Una password è un segreto condiviso dall'entità che svolge il processo di riconoscimento e da quella che deve essere autenticata. Gmail, Facebook, Telnet, FTP e molti altri servizi utilizzano uno schema di autenticazione basato su password. Nel protocollo *ap3.0* Alice invia quindi la sua password a Bob (Figura 8.17).

Dato che l'impiego di password è ampiamente diffuso, potremmo aspettarci che il protocollo *ap3.0* sia piuttosto sicuro. E invece no. Se Trudy intercetta la comunicazione di Alice, può scoprirne la password. Per quanto improbabile possa sembrare, va ricordato che quando si è in collegamento Telnet con un'altra macchina e ci si registra, la password di login è inviata al server in chiaro. Chiunque sia collegato alle LAN di client o server può intercettare (leggere e archiviare) tutti i pacchetti trasmessi sulla LAN e rubare la password di login [Jimenez 1997]. Purtroppo *ap3.0* non esclude questo rischio.

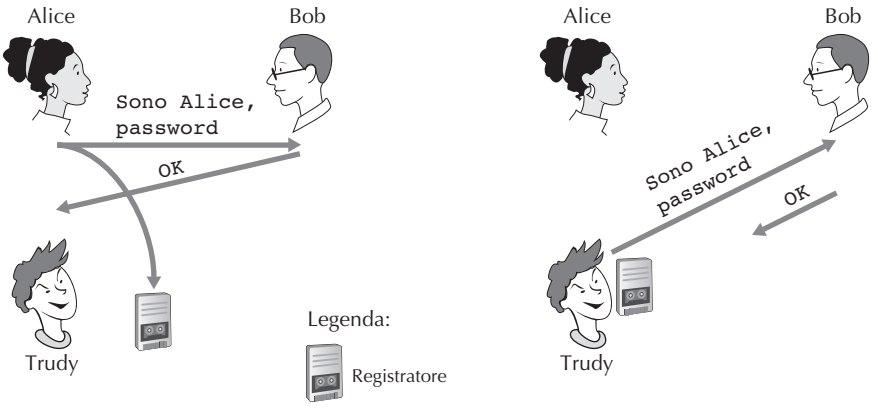

Figura 8.17 Scenario con fallimento del protocollo *ap3.0*.

8.4.4 Protocollo di autenticazione *ap3.1*

Un modo per superare i limiti di *ap3.0* consiste nel cifrare la password in modo da impedire all'intruso di appropriarsene. Se Alice condivide una chiave simmetrica segreta K_{A-B} con Bob, potrebbe utilizzarla per cifrare il suo messaggio di identificazione e la password. Bob provvederebbe quindi a decodificarla e a controllarne la veridicità. Se il responso è positivo, il destinatario si sente sicuro in quanto il mittente, oltre alla password, conosce anche la chiave segreta condivisa con cui è stata cifrata.

Anche se questa versione del protocollo, che chiameremo *ap3.1*, impedisce di scoprire la password di Alice, l'utilizzo della crittografia non risolve il problema dell'autenticazione. Bob è ancora soggetto al cosiddetto **attacco di replica** (*playback attack*). Trudy può infatti inserirsi nella comunicazione, registrare la versione cifrata della chiave e successivamente riprodurla per inviarla a Bob sostenendo di essere Alice. In sostanza, l'impiego della password cifrata non ha reso la situazione diversa da quella del protocollo *ap3.0* (Figura 8.17).

8.4.5 Protocollo di autenticazione *ap4.0*

Lo scenario con fallimento del protocollo della Figura 8.17 deriva dal fatto che Bob non riesce a distinguere fra l'autenticazione originale di Alice e la successiva riproduzione. Vale a dire, Bob non può stabilire se chi sta comunicando con lui sia effettivamente Alice o se i messaggi che stava ricevendo erano una riproduzione registrata. Il lettore molto (ma molto) attento ricorderà che il protocollo dell'handshake a tre vie di TCP deve risolvere lo stesso problema: il lato server non accetta la connessione se il segmento SYN ricevuto è una copia (ritrasmissione) di uno precedentemente inviato. Per fare questo sceglie un numero di sequenza iniziale non utilizzato da molto tempo, lo invia al client e aspetta la risposta con un segmento ACK contenente quel numero. Possiamo adottare la stessa idea per l'autenticazione. Un **nonce**[3] è un numero che il protocollo userà soltanto una volta nel seguente modo.

1. Alice invia il messaggio, "Sono Alice", a Bob.
2. Bob sceglie un nonce, R, e lo trasmette ad Alice.
3. Alice utilizza K_{A-B}, la chiave simmetrica segreta che condivide con Bob, per codificare il nonce, e gli re-invia il valore risultante, $K_{A-B}(R)$. Come nel protocollo *ap3.1*, il punto cruciale è che Alice conosca K_{A-B} e la utilizzi per cifrare un valore che consenta a Bob di riconosce che il messaggio ricevuto proviene proprio da lei. Il nonce viene usato per assicurare che Alice è attiva.
4. Bob decifra il messaggio ricevuto: se il nonce è quello da lui inviato, Alice è autenticata.

---

[3] Il termine è una contrazione dell'espressione "number used once" (numero usato una sola volta) (*N.d.R.*).

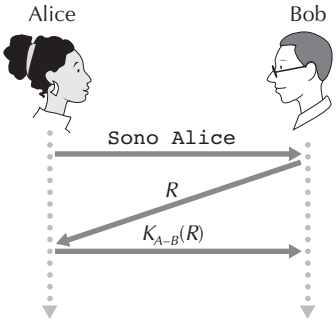

Figura 8.18 Scenario con protocollo *ap4.0*.

Il protocollo *ap4.0* è mostrato nella Figura 8.18. Attraverso l'utilizzo del nonce, *R,* e con il controllo del valore di ritorno $K_{A-B}(R)$, Bob può essere sicuro che si tratta veramente di Alice (perché conosce la chiave segreta necessaria per cifrare *R*) e che si trova in quel momento all'altro capo del collegamento (in quanto ha codificato il nonce, *R*, che Bob ha appena creato).

Per risolvere il problema dell'autenticazione nel protocollo *ap4.0* è stato impiegato un nonce e la crittografia a chiave simmetrica. Una domanda spontanea è se si possa ottenere lo stesso risultato utilizzando la crittografia a chiave pubblica e ovviare al problema del primo scambio della chiave segreta condivisa, come discuteremo in uno dei problemi a fine capitolo.

8.5 Rendere sicura la posta elettronica

Nei paragrafi precedenti abbiamo esaminato le questioni fondamentali relative alla sicurezza di rete, compresa la crittografia simmetrica e a chiave pubblica, l'autenticazione, la distribuzione delle chiavi, l'integrità dei messaggi e la firma digitale. Esaminiamo ora come questi meccanismi siano usati per fornire la sicurezza in Internet.

È interessante vedere come sia possibile fornire servizi di sicurezza in ognuno dei quattro livello superiori dello stack dei protocollo di Internet. Quando la sicurezza viene fornita a uno specifico protocollo a livello di applicazione, l'applicazione che fa uso di quel protocollo usufruirà di uno o più servizi tra riservatezza, autenticazione e controllo di integrità. Quando la sicurezza viene fornita a un protocollo a livello di trasporto, tutte le applicazioni che fanno uso di quel protocollo usufruiranno dei servizi di sicurezza di quel protocollo di trasporto. Quando la sicurezza viene fornita a livello di rete tra due host, tutti i segmenti a livello di trasporto (e quindi anche tutti i dati a livello di applicazione) usufruiscono dei servizi di sicurezza del livello di rete. Quando, infine, la sicurezza viene fornita a livello di collegamento, tutti i frame che attraversano il collegamento usufruiscono dei suoi servizi di sicurezza.

Nei Paragrafi da 8.5 a 8.8 esamineremo come i meccanismi di sicurezza sono impiegati nei livelli applicazione, trasporto, rete e collegamento. Seguendo la struttura

generale di questo libro, iniziamo dalla sommità della pila affrontando il livello di applicazione, con un approccio che utilizza la posta elettronica come caso concreto su cui focalizzare il nostro studio. Ci muoveremo successivamente verso il basso ed esamineremo il protocollo SSL (a livello di trasporto), IPsec (a livello di rete) e la sicurezza nelle LAN wireless IEEE 802.11.

A questo punto, molti di voi si domanderanno perché, in Internet, la funzionalità di sicurezza sia stata fornita a più di un livello e non, semplicemente, solo a quello di rete. A tale quesito ci sono due risposte. In primo luogo, occorre rilevare che, nonostante la sicurezza a livello di rete offra un soddisfacente grado di protezione tramite la cifratura dei dati nei datagrammi (cioè, tutti i segmenti a livello di trasporto) e attraverso l'autenticazione degli indirizzi IP sorgente, tuttavia non può offrire un servizio altrettanto valido a livello utente. Per esempio, un sito di e-commerce non può affidarsi alla sicurezza a livello IP per autenticare un cliente che sta facendo acquisti on-line. Occorre quindi garantire funzionalità di sicurezza più estese nei livelli alti della pila dei protocolli dove, e questo è il secondo aspetto, è in genere più facile sviluppare nuovi servizi Internet, compresi quelli di sicurezza. Per questo, fino a quando la sicurezza non sarà diffusamente implementata a livello di rete (cosa che probabilmente richiederà ancora svariati anni) i progettisti di applicazioni decideranno di installare le funzionalità di sicurezza in base alle loro preferenze. Un classico esempio è costituito da PGP (*pretty good privacy*): un'applicazione client/server per la posta elettronica, tra le prime a essere diffusamente impiegate in Internet, che affronteremo nel prossimo paragrafo.

8.5.1 E-mail sicure

Utilizziamo ora i princìpi di crittografia dei precedenti paragrafi per realizzare, passo dopo passo, in modo incrementale, un progetto ad alto livello per la sicurezza della posta elettronica. Riprendiamo il nostro esempio (Paragrafo 8.1) e immaginiamo che Alice voglia inviare una e-mail a Bob, e che, ancora una volta, Trudy voglia intromettersi.

Iniziamo col definire le caratteristiche di sicurezza cui deve rispondere il nostro progetto. Prima, e più importante, è la **riservatezza**: ovviamente né Alice né Bob vogliono che Trudy legga le loro e-mail. La seconda è l'**autenticazione del mittente**: se Bob riceve il messaggio "Non ti amo più. Non voglio rivederti. La tua ex, Alice", vorrebbe, naturalmente, essere sicuro che il messaggio provenga da Alice e non da Trudy. Un'ulteriore caratteristica è l'**integrità**, cioè la certezza che i messaggi non siano modificati durante il trasporto. Infine, il sistema di posta elettronica dovrebbe fornire l'**autenticazione del ricevente**, vale a dire: Alice vuole essere sicura di aver inviato la lettera proprio a Bob e non a qualcun altro che lo sta impersonando (per esempio Trudy).

Cominciamo con la richiesta più importante: la riservatezza. Il modo più semplice per ottenerla è cifrare il messaggio con una chiave simmetrica (facendo uso, per esempio di DES o AES). Se questa è sufficientemente lunga e se solo i due corrispondenti ne sono in possesso, allora è difficile per chiunque comprendere il messaggio (Paragrafo 8.2). Sebbene il metodo sia semplice, si scontra con la fondamentale difficoltà di distribuire la chiave simmetrica in modo che solo mittente e destinatario ne

abbiano una copia. Così, consideriamo naturalmente un approccio alternativo: la crittografia a chiave pubblica, usando per esempio RSA. Questo approccio vede Bob rendere disponibile la sua chiave pubblica (tramite un server di chiavi pubbliche o sulla sua pagina web); con questa Alice cifra il proprio messaggio che poi invia all'indirizzo e-mail di Bob. Quando Bob riceve il messaggio, lo decifra con la sua chiave privata. Assumendo che Alice sappia con sicurezza che la chiave pubblica è proprio di Bob, e che questa sia sufficientemente lunga, possiamo ritenere di disporre di un eccellente strumento per la segretezza. Purtroppo, la cifratura a chiave pubblica è relativamente inefficiente, in modo particolare per messaggi molto lunghi

Per risolvere il problema dell'efficienza, utilizziamo una chiave di sessione (Paragrafo 8.2.2). In particolare: (1) Alice sceglie arbitrariamente una chiave simmetrica, K_S; (2) cifra il suo messaggio, m, con K_S; (3) codifica la chiave simmetrica con la chiave pubblica di Bob, K_B^+; (4) concatena messaggio cifrato e chiave simmetrica cifrata per formare un "pacco" che (5) invia all'indirizzo e-mail di Bob (Figura 8.19). In questa e nelle figure seguenti, il "+" e "–" cerchiati rappresentano rispettivamente la concatenazione e la separazione di informazioni. Quando Bob riceve il "pacco", (1) utilizza la sua chiave privata K_B^- per ottenere la chiave simmetrica K_S che (2) impiega per decodificare il messaggio m.

Supponiamo, ora, che la principale preoccupazione di Alice e Bob siano l'autenticazione del mittente e l'integrità dei messaggi, ma non siano particolarmente interessati alla sicurezza. Per raggiungere questo obiettivo utilizzano le firme digitali e una funzione di hash (Paragrafo 8.3). Nello specifico, (1) Alice applica una funzione hash H (come MD5) al suo messaggio, m; (2) cifra il risultato con la sua chiave privata, K_A^-, per creare la firma digitale; (3) concatena messaggio in chiaro e firma, (4) e l'invia, come un unico "pacco", all'indirizzo e-mail di Bob. Quando Bob lo riceve, (1) applica la chiave pubblica di Alice, K_A^+, all'hash del messaggio che è stata firmata e (2) confronta il risultato con quanto ottenuto con la sua funzione di hash H (Figura 8.20). Se coincidono, può essere sufficientemente sicuro che il messaggio provenga da Alice e che non sia stato alterato durante il trasferimento (Paragrafo 8.3).

Affrontiamo ora il progetto di un sistema di posta elettronica che risponda alle richieste di segretezza, autenticazione del mittente e integrità dei messaggi: in pratica una combinazione delle procedure descritte nelle Figure 8.19 e 8.20. In questo sce-

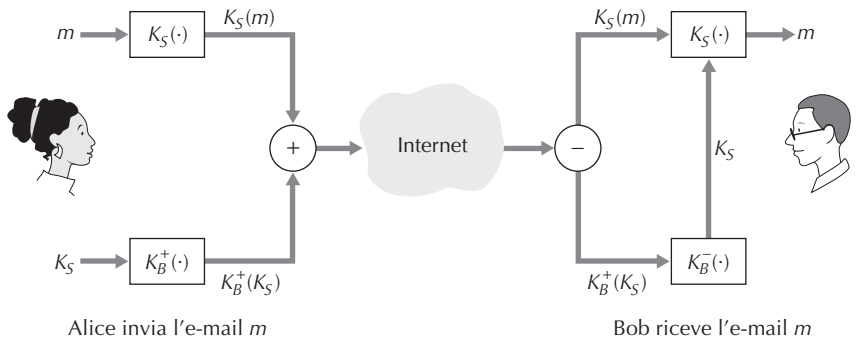

Figura 8.19 Alice utilizza la chiave simmetrica, K_S, per inviare una e-mail segreta a Bob.

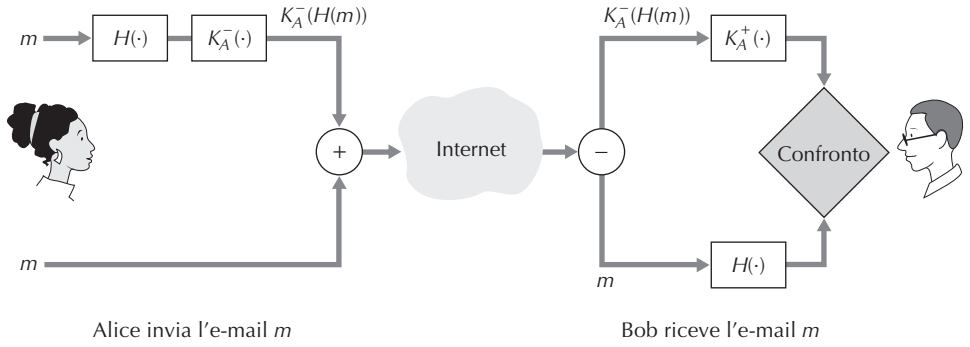

Figura 8.20 Utilizzo di funzioni hash e di firme digitali per l'autenticazione e l'integrità del messaggio.

nario vediamo Alice confezionare una "pacco" preliminare (Figura 8.20), che contiene messaggio originale e hash del messaggio firmata digitalmente, e inviarlo (Figura 8.21). Quando Bob riceve il "pacco", applica la parte che gli compete dello schema della Figura 8.19 e quindi quello della Figura 8.20. Dovrebbe essere chiaro che abbiamo raggiunto il triplice obiettivo: riservatezza, autenticazione del mittente e integrità. Notiamo che in questo schema Alice applica la crittografia a chiave pubblica due volte: prima con la sua chiave privata e poi con la chiave pubblica di Bob. Anche Bob applica due volte la tecnica a chiave pubblica: con la sua chiave privata e con la chiave pubblica di Alice.

Lo schema riportato nella Figura 8.21 fornisce un grado di sicurezza soddisfacente per la maggior parte degli utilizzatori di e-mail. Ma rimane ancora da valutare un argomento importante: occorre che Alice ottenga la chiave pubblica di Bob e questi quella di Alice. Ancora una volta si pone in modo centrale il problema della distribu-

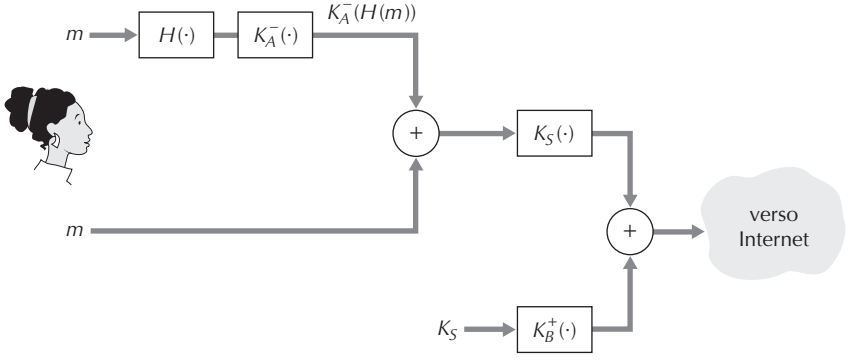

Figura 8.21 Alice utilizza la crittografia a chiave simmetrica, quella a chiave pubblica, una funzione hash e la firma digitale per ottenere segretezza, autenticazione del mittente e integrità del messaggio.

zione. Per esempio, Trudy potrebbe spacciarsi per Bob e dare ad Alice la propria chiave pubblica, riuscendo così a carpire i messaggi inviati a Bob. Un modo sicuro, molto diffuso nella pratica, per affrontare il problema consiste nella certificazione della chiave pubblica per mezzo di una CA, come visto nel Paragrafo 8.3.

8.5.2 PGP

Scritto da Philip R. Zimmermann nel 1991, **pretty good privacy** (**PGP**) è uno schema di cifratura per la posta elettronica diventato uno standard *de facto* e il sito web del progetto distribuisce più di un milione di pagine al mese a utenti di 166 paesi. Esistono svariate versioni di PGP di pubblico dominio: potete trovare la versione per la vostra piattaforma preferita come pure un sacco di documenti interessanti sul sito di riferimento di PGP [PGPI 2016]. L'architettura di PGP è, in sostanza, quella illustrata nella Figura 8.21. A seconda delle versioni il software utilizza: MD5 o SHA per il calcolo della sintesi del messaggio, CAST, triple-DES o IDEA per la crittografia a chiave simmetrica e RSA per quella a chiave pubblica.

Una volta installato, PGP crea una coppia di chiavi: quella pubblica, che può essere collocata sul sito web dell'utente o su un server di chiavi pubbliche, e quella privata, protetta da una password che deve essere inserita ogni volta che viene utilizzata la chiave. Inoltre, il software offre all'utente l'opzione di poter firmare digitalmente il messaggio, di codificarlo o di effettuare entrambe le operazioni. La Figura 8.22 mostra il messaggio con firma PGP che compare come una sottoparte del messaggio di posta. I dati cifrati sono $K_A^-(H(m))$, cioè, la hash del messaggio firmata digitalmente. Come abbiamo già visto, per verificare l'integrità del messaggio, Bob ha bisogno di accedere alla chiave pubblica di Alice.

Osserviamo ora la Figura 8.23 che mostra un messaggio PGP segreto all'interno di una e-mail; naturalmente, il messaggio in chiaro non è incluso. Nel caso in cui il mittente richieda sia riservatezza che integrità, allora PGP conterrà un messaggio come quello della Figura 8.23 all'interno di quello della Figura 8.22.

PGP fornisce anche un meccanismo di certificazione della chiave pubblica, diverso da quello più convenzionale fornito da una CA. Le chiavi pubbliche PGP sono cer-

```
— — -BEGIN PGP SIGNED MESSAGE— — -
Hash: SHA1
Bob:
Possiamo vederci stasera?
Appassionatamente tua, Alice
— — - BEGIN PGP SIGNATURE — — -
Version: PGP for Personal Privacy 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
— — -END PGP SIGNATURE— — -
```

Figura 8.22 Messaggio PGP firmato.

```
— — -BEGIN PGP MESSAGE— — -
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=lKhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
— — -END PGP MESSAGE— — -
```

Figura 8.23 Messaggio PGP segreto.

tificate attraverso una rete di fiducia. Alice stessa può certificare qualsiasi coppia chiave/nome utente quando non ha dubbi sui suoi componenti. Inoltre, PGP permette ad Alice di affermare che si fida di un altro utente per attestare l'autenticità di ulteriori chiavi. Alcuni utenti PGP raccolgono e si scambiano le chiavi pubbliche che certificano vicendevolmente firmandole con le proprie chiavi private.

8.6 Rendere sicure le connessioni TCP: SSL

Nei paragrafi precedenti abbiamo visto come le tecniche crittografiche possano fornire riservatezza, integrità dei dati e autenticazione tra partecipanti a una specifica applicazione: la posta elettronica. In questo paragrafo scenderemo di un livello nella pila di protocolli ed esamineremo come la crittografia può essere usata per arricchire TCP con servizi di sicurezza, comprese la riservatezza, l'integrità dei dati e l'autenticazione end-point. Questa versione arricchita di TCP è comunemente nota come **secure sockets layer** (**SSL**). Una versione leggermente modificata di SSLv3, chiamata **transport layer security** (**TLS**), è stata standardizzata da IETF [RFC 4346].

SSL fu originariamente progettato da Netscape, ma l'idea di base di rendere TCP sicuro è antecedente, si veda per esempio [Woo 1994]. Sin dall'inizio SSL è stato caratterizzato da un uso diffuso. SSL è supportato da tutti i più comuni browser e web server e viene utilizzato da gmail e da tutti i siti di commercio elettronico in Internet (compresi Amazon, eBay, e TaoBao). Centinaia di miliardi di dollari vengono spesi tramite SSL ogni anno. Infatti, se avete mai comperato qualcosa su Internet con la vostra carta di credito, la comunicazione tra il vostro browser e il web server è quasi certamente avvenuta usando SSL: potete accorgervi che viene usato SSL dal vostro browser quando l'URL inizia con https anziché http.

Per comprendere la necessità di SSL consideriamo un tipico scenario di commercio elettronico su Internet. Immaginiamo che Bob, durante la sua navigazione sul Web, si connetta al sito della Alice Incorporated, che vende profumi. Desideroso di fare acquisti, Bob compila una scheda in cui indica la tipologia e la quantità desiderata di profumi, il suo indirizzo e il numero della carta di credito e clicca su "invia". A questo punto non gli resta che attendere di ricevere quanto ordinato (diciamo, attraverso il servizio postale tradizionale) e di verificare l'addebito della spesa sull'estratto conto. Tutto ciò sembra funzionare senza intoppi, ma se non si adottano specifiche misure di sicurezza Bob potrebbe avere qualche brutta sorpresa.

- Se non venisse assicurata la riservatezza (cifratura), un intruso potrebbe intercettare l'ordine, ottenere le informazioni relative alla carta di credito di Bob, e fare acquisti a sue spese.
- Se non venisse assicurata l'integrità dei dati, un intruso potrebbe modificare l'ordine di Bob e fargli comperare, per esempio, 10 volte più bottiglie di profumo di quelle che desidera.
- Infine, se non venisse usata l'autenticazione del server, il sito potrebbe mostrare il logo della Alice Incorporated, ma essere in realtà gestito da Trudy che potrebbe incassare i soldi di Bob e sparire. Oppure, Trudy potrebbe rubare i dati e il numero della carta di credito di Bob e rubare la sua identità.

SSL cerca di risolvere tutti questi problemi aggiungendo a TCP la riservatezza, l'integrità dei dati e l'autenticazione del client e del server.

SSL viene spesso usato per fornire sicurezza alle transazioni che hanno luogo tramite HTTP, tuttavia, dato che SSL rende sicuro TCP, può essere sfruttato da qualsiasi altra applicazione che faccia uso di TCP. SSL fornisce una semplice API (*application programmer interface*) verso le socket, analoga a quella fornita da TCP: quando un'applicazione vuole usare SSL, include le classi/librerie SSL che forniscono l'interfaccia socket SSL allo sviluppatore dell'applicazione. Come mostrato nella Figura 8.24, sebbene SSL risieda tecnicamente nel livello applicativo, dal punto di vista dello sviluppatore è un protocollo di trasporto che fornisce il servizio di TCP arricchito con servizi di sicurezza.

8.6.1 Quadro generale

Iniziamo col descrivere una versione semplificata di SSL, che ci consenta di comprenderne in modo generale le motivazioni e il funzionamento. Faremo riferimento a questa versione semplificata indicandola con "quasi-SSL". Quindi, nel successivo paragrafo, descriveremo SSL nella sua integrità, aggiungendo alcuni dettagli. "Quasi-SSL" e SSL hanno tre fasi: handshake, derivazione delle chiavi e trasferimento dati. Descriveremo queste tre fasi per una sessione di comunicazione tra un client

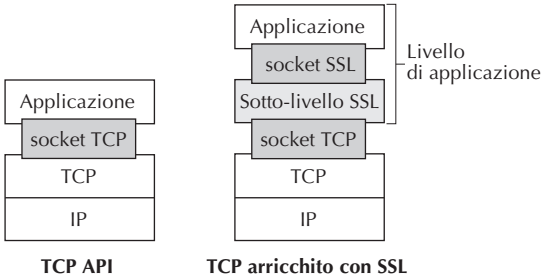

Figura 8.24 Sebbene SSL risieda tecnicamente nel livello applicativo, dal punto di vista dello sviluppatore è un protocollo a livello di trasporto.

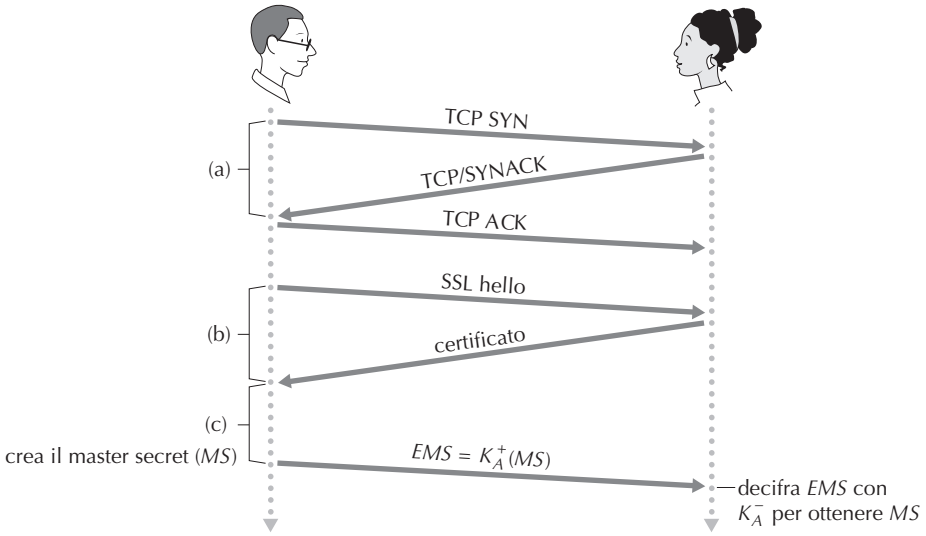

Figura 8.25 Handshake di "quasi-SSL", che inizia con una connessione TCP.

(Bob) e un server (Alice) che dispone di una coppia di chiavi privata/pubblica e di un certificato che lega la sua identità alla sua chiave pubblica.

Handshake

Durante la fase di handshake, Bob ha bisogno di (a) stabilire una connessione TCP con Alice, (b) verificare che Alice sia realmente Alice e (c) inviarle una chiave segreta principale che verrà utilizzata da entrambi per generare tutte le chiavi simmetriche di cui hanno bisogno per la sessione SSL (Figura 8.25). Notate che una volta che la connessione TCP è stata stabilita, Bob manda ad Alice un messaggio "hello" e Alice risponde con il proprio certificato che contiene la sua chiave pubblica. Come abbiamo visto nel Paragrafo 8.3, dato che il certificato è garantito da una CA, Bob sa con certezza che la chiave pubblica del certificato appartiene ad Alice. Bob genera un **master secret** (**MS**): un valore segreto che verrà usato solo per questa sessione SSL e dal quale verranno derivate altre chiavi. Il master secret viene cifrato con la chiave pubblica di Alice, per creare un **encyrpted master secret** (**EMS**) che viene inviato ad Alice, la quale decifra l'EMS con la sua chiave privata e ottiene MS. Dopo questa fase Bob ha autenticato Alice, ed entrambi (ma nessun'altro) conoscono il master secret per questa sessione SSL.

Derivazione delle chiavi

In linea di principio, MS, ora condiviso da Bob e Alice, potrebbe essere usato come la chiave simmetrica di sessione per tutte le successive cifrature e verifiche dell'integrità dei dati. È però generalmente considerato più sicuro che Bob e Alice usino ciascuno chiavi crittografiche differenti, oltre che a impiegare chiavi diverse per la

cifratura e la verifica dell'integrità dei dati. Quindi, Alice e Bob usano MS per generare 4 chiavi:

- E_B = chiave di cifratura di sessione per i dati inviati da Bob ad Alice
- M_B = chiave MAC di sessione per i dati inviati da Bob ad Alice
- E_A = chiave di cifratura di sessione per i dati inviati da Alice a Bob
- M_A = chiave MAC di sessione per i dati inviati da Alice a Bob

Le chiavi possono essere generate semplicemente suddividendo MS in quattro parti (anche se nella reale implementazione di SSL è un po' più complicato, come vedremo fra breve). Alla fine della fase di derivazione delle chiavi, sia Alice sia Bob hanno 4 chiavi. Le due chiavi di cifratura verranno usate per cifrare i dati, e le due chiavi MAC verranno usate per verificarne l'integrità.

Trasferimento dati

Ora che Bob sa con certezza che sta comunicando con Alice, ed entrambi condividono le stesse 4 chiavi di sessione (E_B, M_B, E_A, e M_A), i due possono iniziare a scambiarsi dati in sicurezza sulla connessione TCP. Dato che TCP è un protocollo che si basa su un flusso di byte, un approccio naturale per SSL sarebbe cifrare al volo i dati applicativi e passarli poi a TCP. Ma se facessimo così, dove metteremmo il MAC per la verifica dell'integrità? Certamente non vorremmo aspettare fino alla fine della sessione TCP per verificare l'integrità di tutti i dati di Bob che sono stati inviati. Per risolvere questo problema, SSL suddivide il flusso di dati in record a cui aggiunge un MAC per la verifica dell'integrità e che poi cifra. Per creare il MAC, Bob passa i dati del record assieme alla chiave M_B come a una funzione hash, come spiegato nel Paragrafo 8.3. Per cifrare il pacchetto composto da record e MAC, Bob usa la sua chiave di cifratura di sessione E_B. Il pacchetto cifrato viene poi passato a TCP per essere trasportato in Internet.

Sebbene questo approccio contribuisca a fornire l'integrità dei dati per l'intero flusso del messaggio, non è ancora a prova di bomba. In particolare, supponete che Trudy stia usando un attacco di tipo "man-in-the-middle" e abbia la capacità di inserire, cancellare e sostituire segmenti nel flusso di segmenti TCP inviati da Alice a Bob. Trudy, per esempio, può catturare due segmenti inviati da Bob, invertirne l'ordine, aggiustarne i numeri di sequenza TCP (che non possono essere cifrati) e poi inviare i due segmenti invertiti ad Alice. Assumendo che ciascun segmento TCP incapsuli esattamente un record, diamo uno sguardo a come Alice elaborerà i due segmenti:

1. il TCP di Alice pensa che tutto vada bene e passa i due record al sottolivello SSL.
2. SSL di Alice decifra i due record.
3. SSL di Alice usa il MAC in ciascun record, per verificare l'integrità dei dati.
4. SSL passa i flussi di byte decifrati dei due record al livello applicativo, ma il flusso di byte completo ricevuto da Alice non sarà nell'ordine corretto, a causa dei due record invertiti.

Figura 8.26 Formato del record SSL.

Siete incoraggiati a considerare scenari simili per quando Trudy elimina o sostituisce segmenti.

La soluzione a questo problema, come probabilmente immaginerete, è di usare dei numeri di sequenza oltre che quelli di TCP. SSL si comporta come segue: Bob mantiene un contatore di numeri di sequenza che inizia da zero e viene incrementato per ciascun record che SSL invia. Bob non include effettivamente il numero di sequenza nel record vero e proprio, ma lo include nel MAC, quando ne effettua il calcolo. Quindi, il MAC è ora un hash di dati, cui si aggiunge la chiave MAC M_B e il numero di sequenza corretto. Alice conserva traccia dei numeri di sequenza di Bob, per verificare l'integrità dei dati di un record, includendo nel calcolo del MAC il numero di sequenza appropriato. Questo uso dei numeri di sequenza da parte di SSL evita che Trudy possa eseguire un attacco di tipo man-in-the-middle, come nel caso del riordino e della sostituzione dei segmenti.

Record SSL

Il record SSL (come quello di "quasi-SSL") è mostrato nella Figura 8.26. Il record consiste nei seguenti campi: tipo, versione, lunghezza, dati e MAC. Si noti che i primi tre campi non sono cifrati.

Il campo tipo indica se il record è un messaggio di handshake o uno che contiene dati applicativi. Viene anche usato per chiudere una sessione SSL, come vedremo in seguito. SSL al lato ricevente usa il campo lunghezza per estrarre i record SSL dal flusso di byte TCP in ingresso. Il campo versione è auto esplicativo.

8.6.2 Un quadro più completo

Il paragrafo precedente ha trattato il protocollo "quasi-SSL" e ci è servito per avere una comprensione di base di come funziona SSL e dei motivi per cui viene utilizzato. Con queste conoscenze possiamo scavare un po' più a fondo ed esaminare gli aspetti essenziali dell'effettivo protocollo SSL. In parallelo alla lettura di questa descrizione del protocollo SSL vi sproniamo a completare il laboratorio Wireshark SSL, disponibile sul sito web del libro.

Handshake SSL

SSL non richiede che Alice e Bob usino uno specifico algoritmo a chiave simmetrica o uno specifico a chiave pubblica, o uno specifico MAC, ma consente loro di accordarsi all'inizio della sessione SSL, durante la fase di handshake, su quali algoritmi crittografici useranno. Inoltre, durante la fase di handshake, Alice e Bob si scambiano dei

nonce che vengono usati nella creazione delle chiavi di sessione (E_B, M_B, E_A, e M_A). I passi del reale handshake SSL sono i seguenti.

1. Il client invia, assieme al proprio nonce, la lista degli algoritmi crittografici da lui supportati.
2. Dalla lista, il server sceglie un algoritmo a chiave simmetrica (per esempio, AES), uno a chiave pubblica (per esempio RSA, con una specifica lunghezza di chiave) e un algoritmo MAC. Restituisce al client le proprie scelte, insieme a un certificato e al nonce del server.
3. Il client verifica il certificato, estrae la chiave pubblica del server, genera un pre-master secret (PMS) e lo cifra con la chiave pubblica del server per poi mandarlo cifrato al server.
4. Usando la stessa funzione di derivazione della chiave, come specificato dallo standard SSL, il client e il server calcolano indipendentemente il master secret partendo da PMS e nonce. Il master secret viene poi suddiviso per generare le due chiavi di cifratura e le due chiavi MAC. Inoltre, se l'algoritmo a chiave simmetrica va uso di CBC (come nel caso di 3DES e AES) i due vettori di inizializzazione (uno per ogni capo della connessione) sono anch'essi ottenuti dal master secret. D'ora in poi tutti i messaggi inviati tra il client e il server sono cifrati e autenticati con il MAC.
5. Il client invia un MAC di tutti i messaggi di handshake.
6. Il server manda un MAC di tutti i messaggi di handshake.

Gli ultimi due passi proteggono l'handshake dalla manomissione. Per accorgersene, si osservi che nel Passo 1, il client offre tipicamente una lista di algoritmi, alcuni forti e alcuni deboli. Questa lista di algoritmi viene inviata in chiaro, in quanto né gli algoritmi di cifratura né le chiavi sono state ancora accordate. Trudy, se usa un approccio di tipo man-in-the-middle, potrebbe cancellare dalla lista gli algoritmi più sicuri, costringendo il client a selezionare un algoritmo debole. Per evitare questo tipo di attacco di alterazione, nel Passo 5 il client fa pervenire un MAC della concatenazione di tutti i messaggi di handshake, che ha spedito e ricevuto. Il server può confrontare questo MAC con quello dei messaggi di handshake da lui mandati e ricevuti: se c'è un'inconsistenza il server può chiudere la connessione. In modo analogo, il server invia un MAC dei messaggi di handshake che ha visto, consentendo al client di verificarne la inconsistenza.

Potreste stupirvi dell'esistenza delle nonce ai passi 1 e 2. I numeri in sequenza non sono sufficienti a prevenire gli attacchi basati sulla ripetizione dei segmenti? La risposta è sì, ma da soli non riescono a prevenire quelli basati sulla ripetizione delle connessioni. Supponiamo, per esempio, che Trudy spii tutti i messaggi tra Alice e Bob. Il giorno successivo Trudy finge di essere Bob e invia ad Alice esattamente la stessa sequenza di messaggi che Bob le aveva inviato il giorno precedente. Se Alice non usa un nonce risponde esattamente con la stessa sequenza di messaggi che ha inviato il giorno prima. Alice non avrà alcun sospetto perché ogni messaggio che riceve supera il controllo di integrità. Se Alice fosse un server di e-commerce, penserebbe che Bob

sta facendo un secondo ordine esattamente per la stessa cosa. D'altra parte, includendo una nonce nel protocollo, Alice invia nonce diverse su ogni sessione TCP, forzando le chiavi di cifratura a essere diverse nelle due occasioni. Perciò, quando Alice riceve i record SSL replicati da Trudy, questi falliscono i controlli di integrità e quindi la transazione non avrà successo. Riassumendo, in SSL, si usano i nonce per difesa contro la ripetizione delle connessioni (*connection reply attack*) e i numeri di sequenza per difendersi contro il rinvio di pacchetti individuali durante una sessione aperta.

Chiusura della connessione

A un certo punto, tanto Bob quanto Alice vorranno terminare la sessione SSL. Un approccio potrebbe essere quello di lasciare che sia Bob a concludere la sessione, semplicemente terminando la connessione TCP sottostante, cioè Bob invia un segmento TCP FIN ad Alice. Ma un modello così semplice prepara il campo a un attacco di tipo *truncation*, dove Trudy ancora un volta si pone nel mezzo di una sessione SSL e la termina prematuramente con un messaggio TCP FIN. Se Trudy facesse questo, Alice penserebbe di aver ricevuto tutti i dati da Bob, quando effettivamente ne ha ricevuto solo una parte. La soluzione a questo problema è indicare nel campo tipo se il record serve a terminare la sessione SSL. Sebbene il tipo SSL venga inviato in chiaro, è autenticato dal ricevente usando il MAC del record. Includendo questo tipo di campo, se Alice dovesse ricevere un TCP FIN prima di aver ricevuto il record SSL di chiusura, saprebbe che sta accadendo qualcosa di insolito.

Così si completa la nostra introduzione di SSL. Abbiamo visto che SSL impiega molti dei princìpi di crittografia discussi nei Paragrafi 8.2 e 8.3. I lettori che vogliono esplorare più in profondità SSL possono leggere il libro molto interessante su SSL di Rescorla [Rescorla 2001].

8.7 Sicurezza a livello di rete: IPsec e reti private virtuali

Comunemente conosciuto come **IPsec**, il protocollo di sicurezza di IP fornisce sicurezza a livello di rete. IPsec rende sicuri i datagrammi IP tra due entità come host e router. Come descriveremo tra poco, molti istituti (aziende, enti governativi, organizzazioni non-profit, e così via) usano IPsec per creare **reti private virtuali** (VPN, *virtual private network*) che fanno uso della Internet pubblica.

Prima di affrontare le specifiche di IPsec facciamo, però, qualche passo indietro e consideriamo che cosa significa fornire riservatezza a livello di rete. Per ottenere riservatezza tra due entità a livello di rete (siano esse una coppia di host, una coppia di router o un host e un router) l'entità che invia cifra il campo dati di tutti i pacchetti in partenza verso l'entità che riceve. Il campo dati potrebbe essere costituito da un segmento TCP o UDP, oppure da un messaggio ICMP, e così via. Se fosse disponibile un simile servizio, allora tutti i dati inviati da una entità all'altra (comprese e-mail, pagine web, messaggi di handshake TCP e messaggi di gestione come ICMP e SNMP) sarebbero occultati a eventuali terze parti che stanno intercettando traffico di rete.

Oltre alla riservatezza, un protocollo di sicurezza a livello rete potrebbe potenzialmente fornire altri livelli di sicurezza. Per esempio, potrebbe fornire il servizio di autenticazione della sorgente, in modo che l'entità ricevente possa verificare la sorgente del datagramma. Potrebbe inoltre fornire servizi di integrità dei dati in modo che l'entità ricevente possa controllare se il datagramma sia stato falsificato durante la trasmissione. Un altro servizio che potrebbe fornire è la prevenzione agli attacchi basati su repliche facendo in modo che Bob possa rilevare i datagrammi duplicati inseriti durante un attacco. Vedremo presto che IPsec fornisce effettivamente meccanismi per tutti questi tipi di servizi di sicurezza: riservatezza, autenticazione della sorgente, integrità dei dati e prevenzione degli attacchi di replay.

8.7.1 IPsec e le reti virtuali private (VPN)

Un istituto con sedi in più aree geografiche distinte desidera spesso una propria rete IP, in modo che i suoi host e server possano comunicare in modo sicuro e riservato. Per conseguire questo obiettivo, l'istituto potrebbe effettivamente installare una rete fisica indipendente e separata dalla Internet pubblica, comprensiva di router, collegamenti e infrastruttura DNS. Una rete indipendente di questo tipo, dedicata a una istituzione particolare, è chiamata **rete privata**. Ovviamente i costi di acquisto, installazione e manutenzione sono molto elevati.

Oggigiorno molti istituti, invece di installare e mantenere una rete privata, creano una VPN sulla Internet pubblica, nel senso che il traffico tra uffici viene inviato su questa piuttosto che su una rete fisica indipendente. Per fornire riservatezza, il traffico tra uffici viene cifrato prima di essere inviato. Nella Figura 8.27 è mostrato un semplice esempio di una VPN di una istituzione che consiste di un quartier generale, di

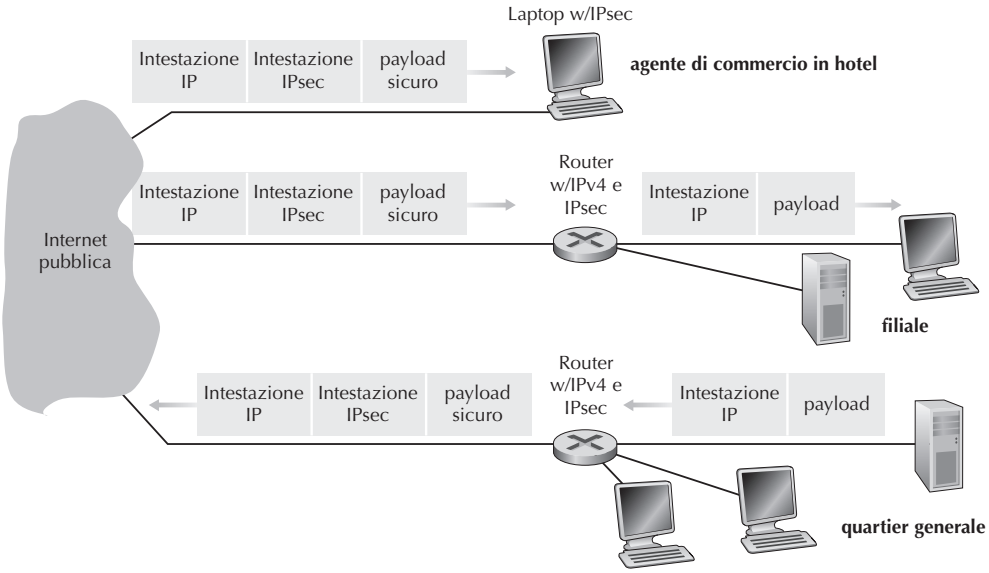

Figura 8.27 Rete privata virtuale (VPN).

una filiale e di agenti di commercio che generalmente accedono a Internet dalle loro camere di albergo (nella figura è rappresentato un solo agente di commercio). In questa VPN due host all'interno del quartier generale o nella filiale si scambiano datagrammi IP, usano il caro vecchio IPv4, quindi senza servizi IPsec. Tuttavia, quando due host comunicano su un percorso che attraversa Internet, allora cifrano il traffico prima che questo entri in Internet.

Vediamo un semplice esempio del funzionamento di una VPN nel contesto della Figura 8.27. Quando un host nel quartier generale invia un datagramma IP a un agente di commercio che si trova in un hotel, il gateway del quartier generale converte il datagramma IPv4 in un datagramma IPsec che viene inoltrato attraverso Internet. Poiché il datagramma IPsec ha a tutti gli effetti una intestazione IPv4 tradizionale, i router lo elaborano come se fosse un datagramma IPv4 ordinario. Ma, come mostrato nella Figura 8.27, il payload del datagramma IPsec include un'intestazione IPsec, usata per l'elaborazione IPsec; inoltre, il payload di un datagramma IPsec è cifrato. Quando il datagramma IPsec arriva al laptop dell'agente di commercio, il suo sistema operativo decripta il payload (e fornisce altri servizi di sicurezza, come la verifica dell'integrità dei dati) e passa i dati decifrati al protocollo di livello superiore, quale TCP o UDP.

Fin qui abbiamo dato solo una visione generale di come una istituzione possa utilizzare IPsec per creare una VPN; vediamone ora i dettagli.

8.7.2 I protocolli AH e ESP

IPsec è un animale molto complesso, definito in più di una dozzina di RFC, tra i quali RFC 4301, che descrive l'architettura complessiva di sicurezza e l'RFC 6071, che fornisce una panoramica della suite dei protocolli IPsec. In questo libro affronteremo tali protocolli con un approccio operativo e pedagogico.

Due sono i principali protocolli della suite IPsec: il protocollo di *authentication header* (AH, intestazione per l'autenticazione) e l'*encapsulation security payload* (ESP, incapsulamento sicuro del payload). Un'entità sorgente IPsec, quando invia datagrammi sicuri a un'entità di destinazione usa il protocollo AH o il protocollo ESP. Il protocollo AH fornisce solo l'autenticazione della sorgente e l'integrità dei dati, mentre ESP fornisce anche la riservatezza. Poiché la riservatezza è spesso un fattore critico per le VPN e le altre applicazioni IPsec, il protocollo ESP è molto più diffuso di quello AH e in questa sede ci focalizzeremo esclusivamente su di esso.

8.7.3 Associazioni di sicurezza

Prima di procedere all'invio di datagrammi sicuri, l'host sorgente e quello di destinazione creano un canale logico a livello di rete, chiamato **associazione di sicurezza**, (SA, *security association*). Essendo però le SA unidirezionali, se gli host vogliono scambiarsi datagrammi sicuri sono richieste due SA, cioè due connessioni logiche, una in ciascuna direzione.

Consideriamo come esempio, ancora una volta, la VPN raffigurata nella Figura 8.27 di un'istituzione che consiste di un quartier generale, di una filiale e di *n* agenti di commercio. Supponiamo inoltre che il traffico IPsec tra il quartier generale e la fi-

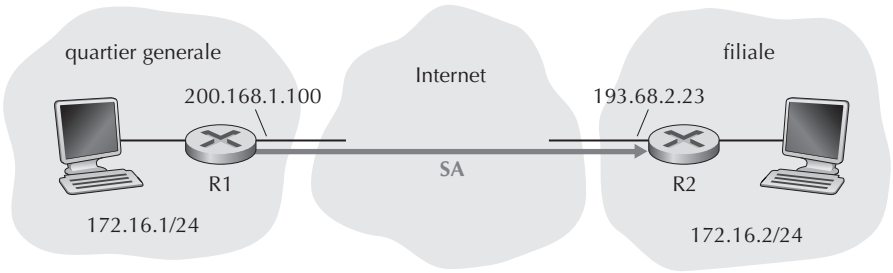

Figura 8.28 Associazione di sicurezza (SA) da R1 a R2.

liale e quello tra il quartier generale e gli agenti di commercio siano bidirezionali. In questa VPN, quante SA esistono? Per rispondere a questa domanda, notate che esistono due SA tra il gateway del quartier generale e quello della filiale, una per ogni direzione; esistono inoltre due SA tra ogni laptop e il gateway del quartier generale, una per ogni direzione. Quindi in totale esistono $(2 + 2n)$ SA. Non dimenticate tuttavia che non tutto il traffico inviato in Internet dai gateway o dai laptop viene reso sicuro tramite IPsec. Se, per esempio, un host nel quartier generale accedesse a un web server, come Amazon o Google su Internet, il gateway e i laptop immetterebbero in Internet sia datagrammi IPv4 sia datagrammi IPsec.

Per guardare come è fatta all'interno una SA riferiamoci a quella tra il router R1 e il router R2 della Figura 8.28, immaginando che il primo sia il gateway del quartier generale e il secondo quello della filiale della Figura 8.27. Il router R1 mantiene informazioni di stato sulla SA, tra cui:

- un identificatore a 32 bit della SA, chiamato **indice dei parametri di sicurezza** (SPI, *security parameter index*)
- l'interfaccia di origine della SA, in questo caso 200.168.1.100 e l'interfaccia di destinazione della SA, in questo caso 193.68.2.23
- il tipo di codifica utilizzato; per esempio 3DES con CBC
- la chiave di codifica
- il tipo di controllo di integrità; per esempio HMAC con MD5
- la chiave di autenticazione.

Il router R1, ogni volta che costruisce un datagramma IPsec da inoltrare su questa SA, accede a queste informazioni di stato per determinare come autenticare e cifrare il datagramma. Analogamente, il router R2 mantiene le stesse informazioni di stato per questa SA per autenticare e decifrare tutti i datagrammi IPsec in arrivo da questa SA.

Spesso un'entità IPsec, router o host, mantiene le informazioni di stato per molte SA. Per esempio, il gateway del quartier generale della VPN mostrata nella Figura 8.27 mantiene le informazioni di stato di $(2 + 2n)$ SA. Le entità IPsec memorizzano le informazioni di stato di tutte le loro SA nel loro **database di associazione di sicurezza** (SAD, *security association database*): una struttura dati che risiede nel kernel del sistema operativo.

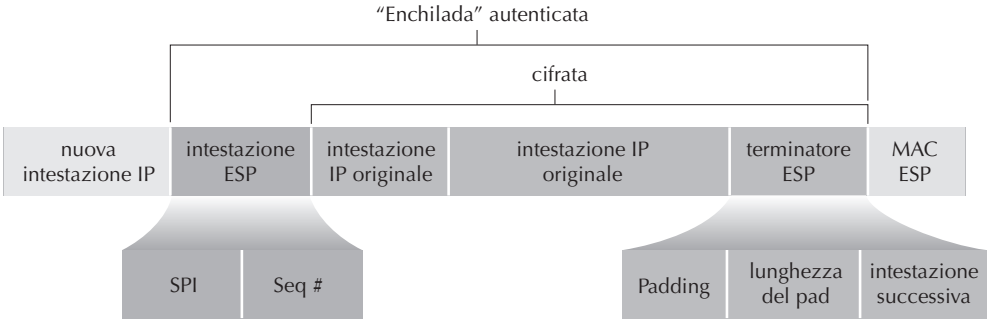

Figura 8.29 Formato dei datagrammi IPsec.

8.7.4 Il datagramma IPsec

Dopo aver descritto le SA, passiamo ora ai datagrammi IPSec. IPsec prevede due forme diverse di pacchetto: quella chiamata **modalità tunnel** e quella chiamata **modalità trasporto**, di cui la prima, essendo più appropriata per le VPN, è più utilizzata. Ancora una volta, al fine di semplificare la trattazione ci focalizzeremo esclusivamente sulla modalità tunnel, appresa la quale si può affrontare facilmente da soli la modalità trasporto.

Il formato del datagramma IPsec è mostrato nella Figura 8.29. Esaminiamo i campi IPsec nel contesto della Figura 8.28. Supponiamo che il router R1 riceva un datagramma IPv4 dall'host 172.16.1.17 nella rete del quartier generale destinato all'host 172.16.2.48 nella rete della filiale. Il router R1, per convertire il datagramma IPv4 in un datagramma IPsec, usa la seguente ricetta:

- appende in fondo al datagramma IPv4 (che comprende i campi di intestazione originari) un campo "coda ESP" cifra il risultato usando l'algoritmo e la chiave specificati dalla SA
- appende all'inizio dei dati cifrati un campo chiamato "intestazione ESP"; il pacchetto risultante è chiamato "enchilada"[4] crea un'autenticazione MAC sull'enchilada usando l'algoritmo e la chiave specificati nella SA
- appende il MAC alla fine dell'enchilada e ottiene il payload
- infine, crea una nuova intestazione IP con tutti i campi di intestazione classici di IPv4, lunghi normalmente 20 byte e la appende prima del payload calcolato al punto precedente.

[4] Dal nome di un piatto tipico messicano (*N.d.R.*).

Si noti che il datagramma IPsec risultante è un datagramma IPv4 genuino, formato dai campi di intestazione tradizionali di IPv4 seguiti dal payload. Però, in questo caso, il payload contiene un'intestazione ESP, il datagramma originario IP, una coda ESP e un campo di autenticazione ESP (con il datagramma originario e la coda ESP cifrati). Il datagramma IP originale ha come indirizzo IP sorgente 172.16.1.17 e come indirizzo IP di destinazione 172.16.2.48. Tali indirizzi sono inclusi e cifrati come parte del payload del datagramma IPsec, ma che cosa succede per quanto riguarda gli indirizzi IP sorgente e destinazione che stanno nella nuova intestazione IP, vale a dire nella intestazione più a sinistra del datagramma IPsec? Come potevate aspettarvi, hanno i valori delle interfacce dei router sorgente e destinazione ai capi del tunnel: 200.168.1.100 e 193.68.2.23. Inoltre il numero di protocollo nel nuovo campo di intestazione IPv4 non è TCP, UDP o SMTP, ma 50, indicando che è un datagramma IPsec che usa il protocollo ESP.

Il datagramma IPsec, dopo essere stato inviato da R1 sulla Internet pubblica, passerà attraverso molti router prima di giungere a R2. Tutti questi router elaboreranno il datagramma come se fosse un datagramma ordinario, in quanto non sono consapevoli che esso trasporti dati cifrati con IPsec. Dal punto di vista di tali router la destinazione ultima del datagramma è R2, in quanto l'indirizzo IP di destinazione nell'intestazione più esterna è R2.

Abbiamo appena visto come viene costruito un datagramma IPsec; analizziamo ora gli ingredienti dell'enchilada. Vediamo nella Figura 8.29 che la coda ESP consiste di tre campi: padding; lunghezza del pad e intestazione successiva. Ricordate che i cifrari a blocchi richiedono che il messaggio da cifrare sia un multiplo intero della lunghezza del blocco. Il padding, che consiste di byte privi di significato, viene aggiunto al datagramma originario, insieme ai campi lunghezza del pad e intestazione successiva, in modo che il messaggio risultante sia un numero intero di blocchi. Il campo lunghezza del pad indica all'entità ricevente quanti byte di padding sono stati inseriti e che quindi devono essere rimossi. Il campo intestazione successiva identifica il tipo (per esempio, UDP) di dati contenuti nel campo payload. Il payload (tipicamente costituito dal datagramma IP originale) e la coda ESP vengono concatenati e quindi cifrati. L'intestazione ESP, appesa all'inizio di questa unità cifrata, viene inviata in chiaro e consiste di due campi: l'SPI e il numero di sequenza. Il primo indica all'entità ricevente a quale SA il datagramma appartiene; l'entità ricevente può quindi usare l'SPI come indice per il suo SAD e determinare gli algoritmi e le chiavi di autenticazione e decifratura appropriate. Il campo numero di sequenza è usato per difesa contro gli attacchi basati su ripetizione.

L'entità mittente appende anche un MAC di autenticazione. Come detto precedentemente, essa calcola il MAC sull'intera enchilada, che consiste di un'intestazione ESP, del datagramma IP originario e della coda ESP, di cui gli ultimi due cifrati. Ricordiamo che il mittente, per calcolare un MAC, appende una chiave MAC segreta all'enchilada e quindi calcola una hash di lunghezza fissata del risultato.

R2, quando riceve il datagramma IPsec, osserva che l'indirizzo IP di destinazione del datagramma è R2 stesso, che quindi elabora il datagramma. R2, poiché il campo

protocollo nell'intestazione IP più a sinistra è 50, applica l'elaborazione IPsec ESP al datagramma. Per prima cosa, guardando nell'enchilada, usa l'SPI per determinare a quale SA il datagramma appartiene. Successivamente, calcola il MAC dell'enchilada e verifica che sia consistente con il valore del campo MAC ESP. In caso affermativo sa che l'enchilada proviene da R1 e non è stata falsificata. Il terzo passo consiste nel controllo del campo numero di sequenza per verificare che il datagramma sia nuovo e non replicato. Il quarto passo consiste nella decifratura della parte cifrata usando l'algoritmo e la chiave associati alla SA. Quinto, rimuove il padding ed estrae il datagramma IP originale. Infine, inoltra il datagramma originale nella rete della filiale verso la destinazione. Whew, una ricetta complicata? Beh, nessuno ha mai detto che preparare e srotolare una enchilada fosse semplice.

In effetti c'è un'altra importante sottigliezza da affrontare, e riguarda la seguente domanda: R1, quando riceve un datagramma non sicuro da un host nel quartier generale destinato a un indirizzo IP di destinazione esterno, come fa a sapere se debba essere convertito in un datagramma IPsec? E se deve essere elaborato da IPsec, come fa R1 a sapere quale, o quali, SA debbano essere usate per costruire il datagramma IPsec? Il problema viene risolto facendo in modo che le entità IPsec, insieme a un SAD, mantenga un'altra struttura dati chiamata **database delle regole di sicurezza** (SPD, *security policy database*). L'SPD indica che tipi di datagrammi debbano essere elaborati con IPsec in funzione degli indirizzi IP di sorgente e destinazione e del tipo di protocollo; per quelli da elaborare, indica quale SA debba essere utilizzata. In un certo senso le informazioni contenute nell'SPD indicano che cosa fare quando un datagramma arriva, quelle nel SAD indicano come farlo.

Riepilogo dei servizi IPsec

Infine, quali servizi fornisce esattamente IPsec? Esaminiamoli dal punto di vista di un attaccante, per esempio Trudy, che sta sul percorso tra R1 e R2 nella Figura 8.28. Assumiamo nella nostra discussione che Trudy non conosca le chiavi di autenticazione e cifratura usate dalla SA. Che cosa può fare Trudy e che cosa non può fare? In primo luogo, Trudy non può vedere il datagramma originale. Infatti, non solo i dati del datagramma originale sono nascosti a Trudy, ma anche il numero di protocollo, l'indirizzo IP sorgente e l'indirizzo IP di destinazione. Dei datagrammi inviati sulla SA Trudy può solo sapere che sono originati da un host in 172.16.1.0/24 e destinati a un host in 172.16.2.0/24. Non sa che trasportano dati TCP, UDP o ICMP e nemmeno se trasportano dati HTTP, SMTP o di altro tipo. Questo tipo di riservatezza va oltre quella fornita da SSL. In secondo luogo, supponiamo che Trudy tenti di falsificare un datagramma nella SA invertendone alcuni bit. Quando il datagramma falsificato arriva a R2, il controllo di integrità tramite MAC fallirà, frustrando i tentativi di Trudy. In terzo luogo supponiamo che Trudy tenti di mascherarsi da R1, creando un datagramma IPsec con sorgente 200.168.1.100 e destinazione 193.68.2.23. Tale attacco di Trudy sarà inutile in quanto il datagramma arrivato a R2 fallirà il controllo di integrità. Infine, includendo IPsec i numeri di sequenza, Trudy non sarà in grado di sferrare con successo un attacco basato sulla ripetizione. Riassumendo, come proclamato

all'inizio del paragrafo, IPsec fornisce, tra qualunque coppia di dispositivi che elaborano pacchetti a livello di rete, riservatezza, autenticazione della sorgente, integrità dei dati e prevenzione degli attacchi basati su ripetizione.

8.7.5 IKE: gestione delle chiavi in IPsec

Quando una VPN ha pochi punti terminali (per esempio, solo due router come nella Figura 8.28), l'amministratore di rete può inserire manualmente le informazioni delle SA (algoritmi e chiavi di cifratura e autenticazione e l'ESP) nel SAD dei router. Questa procedura manuale è chiaramente impraticabile per grandi VPN con centinaia o anche migliaia di router e host IPsec per le quali sono necessari meccanismi automatici di creazione delle SA. IPsec fa tutto ciò tramite il protocollo IKE, Internet Key Exchange, specificato nell'RFC 5996.

IKE è in parte simile all'handshake di SSL (Paragrafo 8.6). Ogni entità IPsec ha un certificato che include la chiave pubblica dell'entità. Come in SSL, il protocollo IKE ha due certificati, negozia gli algoritmi di autenticazione e cifratura e scambia in modo sicuro il materiale per creare chiavi di sessione nelle SA IPsec. Al contrario di SSL, IKE impiega due fasi per eseguire questi compiti.

Vediamo tali fasi nel contesto dei due router, R1 e R2, nella Figura 8.28. La prima fase consiste nei seguenti due scambi di coppie di messaggi tra R1 e R2.

- Durante il primo scambio di messaggi, i due lati usano Diffie-Hellman (si vedano i problemi a fine capitolo) per creare una **SA IKE** bidirezionale tra i router. Tanto per aumentare la confusione, questa SA IKE bidirezionale è completamente diversa dalla SA IPsec discussa nei Paragrafi 8.6.3 e 8.6.4. La SA IKE fornisce un canale di autenticazione e cifratura tra i router. Durante il primo scambio di messaggi, vengono stabilite le chiavi per la cifratura e autenticazione della SA IKE e un master secret che verrà utilizzato nella fase 2 per calcolare le chiavi SA IPsec. Si osservi che durante questo primo passo le chiavi pubbliche e private RSA non vengono utilizzate. In particolare, né R1 né R2 rivelano la propria identità firmando un messaggio con la propria chiave privata.
- Durante il secondo scambio di messaggi, i due lati si rivelano le loro identità firmando i loro messaggi. Tuttavia, poiché i messaggi sono inviati su un canale SA IKE sicuro, le identità non vengono rivelate a una spia passiva. Inoltre, durante questa fase, i due lati negoziano gli algoritmi di cifratura e di autenticazione IPsec che devono essere utilizzati dalla SA IPsec.

Nella fase 2 di IKE i due lati creano una SA per ogni direzione. Alla fine della fase le chiavi di sessione per l'autenticazione e la cifratura sono stabilite da entrambe le parti per le due SA. Le due parti possono quindi utilizzare l'SA per inviare datagrammi sicuri, come descritto nei Paragrafi 8.7.3 e 8.7.4. La motivazione principale delle due fasi di IKE è il costo computazionale; poiché la seconda fase non richiede alcuna crittografia a chiave pubblica, IKE può generare un gran numero di SA tra le due entità IPsec con un costo computazionale relativamente basso.

8.8 Sicurezza nelle LAN wireless

Nelle reti wireless, in cui le onde radio che trasportano i frame possono propagarsi oltre gli edifici che contengono le stazioni base e i terminali, la sicurezza assume una particolare importanza. In questo paragrafo presentiamo una breve introduzione alla sicurezza wireless. Per una trattazione più dettagliata si veda il testo molto interessante di Edney e Arbaugh [Edney 2003].

Il problema della sicurezza nelle reti di tipo 802.11 ha attratto l'attenzione sia degli ambienti tecnici sia della stampa, in quanto è ormai ampiamente riconosciuto che la specifica 802.11 originale contiene non solo considerevoli difetti, ma che è anche possibile scaricare software di pubblico dominio che rende vulnerabile chi ne utilizza i meccanismi di sicurezza.

Nel seguente paragrafo analizzeremo i meccanismi di sicurezza inizialmente standardizzati nella specifica 802.11, universalmente noti come **WEP** (**wired equivalent privacy**), pensato, come dice il nome, per assicurare un livello di sicurezza simile a quello delle reti cablate. Tratteremo poi qualche lacuna di sicurezza di WEP ed esamineremo lo standard 802.11i, una versione più sicura di 802.11 adottata nel 2004.

8.8.1 Wired equivalent privacy (WEP)

Il protocollo WEP di IEEE 802.11 fornisce autenticazione e codifica dei dati tra terminale e access point wireless (cioè, una stazione base), con un approccio a chiave simmetrica condivisa. Non specifica però l'algoritmo di gestione delle chiavi, per cui si presume che host e stazione base si debbano accordare con una procedura fuori banda. L'autenticazione è effettuata come segue.

1. Un terminale wireless richiede l'autenticazione (da parte di un punto di accesso).
2. Il punto di accesso risponde alla richiesta inviando un nonce a 128 byte.
3. L'host codifica il nonce con la chiave simmetrica condivisa e lo ritrasmette.
4. Il punto di accesso decifra il nonce.

Se il nonce decifrato corrisponde a quello inviato all'host, allora l'host è autenticato dall'access point.

L'algoritmo di cifratura dei dati WEP è illustrato nella Figura 8.30. Si assume che terminale e access point dispongano di una chiave segreta simmetrica a 40 bit, K_S, cui viene aggiunto un vettore di inizializzazione (IV) di 24 bit, creando così una chiave di 64 bit che sarà utilizzata per codificare ciascun frame. Dato che IV cambia da un frame all'altro, ciascun frame sarà cifrato con una diversa chiave a 64 bit. La cifratura è effettuata come segue. Per prima cosa viene valutato un CRC (Paragrafo 5.2) di 4 byte per il payload. Il payload e i quattro byte di CRC sono quindi codificati col cifrario a flusso RC4. Per maggiori dettagli su questo metodo si può consultare [Schneier 1995] e [Edney 2003]; per i nostri scopi è sufficiente sapere che, a partire da una chiave, in questo caso, i 64 bit (K_S, IV), l'algoritmo RC4 ne produce una serie, k_1^{IV}, k_2^{IV}, k_3^{IV} …, ciascuna delle quali viene utilizzata per cifrare i dati e il valore del

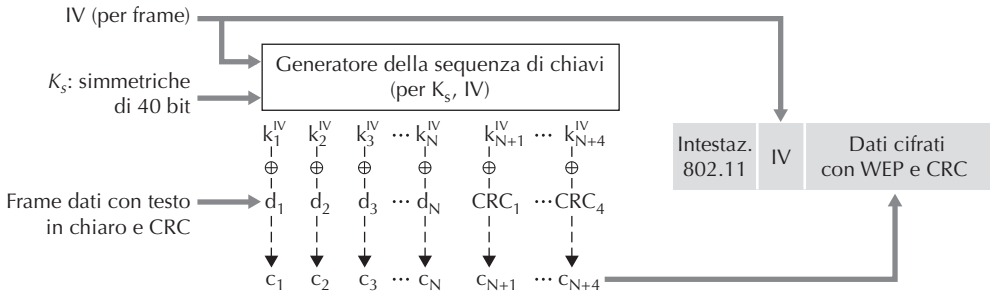

Figura 8.30 Protocollo WEP 802.11.

CRC nei frame. Ai fini pratici, possiamo supporre che queste operazioni siano effettuate un byte alla volta. La cifratura esegue lo XOR tra il byte dei dati, d_i, e la chiave *i*-esima, K_i^{IV}, nel flusso dei valori di chiave generati da (K_S, *IV*) per ottenere il byte del testo cifrato, c_i:

$$c_i = d_i \oplus k_i^{IV}$$

Il valore di IV cambia da un frame all'altro e viene inserito in chiaro nell'intestazione di ogni frame 802.11 cifrato con WEP (Figura 8.32). Il ricevente prende la chiave segreta simmetrica a 40 bit, che condivide con il mittente, aggiunge i 24 bit di IV, e utilizza la stringa risultante (che è identica alla chiave utilizzata dal mittente) per decifrare il frame.

$$d_i = c_i \oplus k_i^{IV}$$

Un uso ottimale dell'algoritmo RC4 richiede che le chiavi di 64 bit (che cambiano regolarmente a ogni frame) non siano *mai* riutilizzate. Per una data K_S (che varia raramente, o per nulla), ci sono 2^{24} chiavi univoche. Se queste sono individuate in modo casuale, la probabilità di scegliere lo stesso valore di IV (e quindi utilizzare la stessa chiave a 64 bit) supera il 99% dopo 12.000 frame. Con dimensioni di pacchetto di 1 Kbyte e frequenza trasmissiva di 11 Mbps, sono sufficienti pochi secondi per la loro trasmissione. Inoltre, dato che IV è trasmesso in chiaro, un intruso saprà quando viene utilizzato un suo valore duplicato.

Per evidenziare i problemi che possono verificarsi quando si utilizza una chiave duplicata, consideriamo il seguente attacco di testo in chiaro scelto, portato da Trudy contro Alice. Supponiamo che Trudy (magari utilizzando IP spoofing) invii una richiesta (HTTP o FTP) ad Alice per trasmettere un file di cui conosce il contenuto d_1, d_2, d_3, d_4 …. Trudy, ovviamente osserva anche i dati cifrati c_1, c_2, c_3, c_4 …. Se effettuiamo l'OR esclusivo di c_i con i membri dell'uguaglianza $d_i = c_i \oplus k_i^{IV}$ otteniamo:

$$d_i \oplus c_i = k_i^{IV}$$

Quindi, può utilizzare i valori d_i e c_i per calcolare k_i^{IV} e la volta successiva che si accorge che viene utilizzato lo stesso valore di IV, conoscendo la sequenza, k_1^{IV}, k_2^{IV}, k_3^{IV}, …, sarà in grado di decifrare il messaggio.

WEP presenta molti altri problemi di sicurezza. La descrizione di un attacco che sfrutta un difetto di RC4 quando si scelgono chiavi inappropriate si trova in [Fluhrer 2001], mentre una trattazione di come implementare e sfruttare in modo efficiente questo attacco è reperibile in [Stubblefield 2002]. Un altro aspetto riguarda i bit CRC (Figura 8.32) trasmessi nel pacchetto 802.11 per rilevare i bit alterati nel payload. Infatti, un attaccante può variare il contenuto cifrato, calcolare il CRC del testo modificato e introdurlo nel frame WEP che, purtroppo, sarà ritenuto valido dal ricevente. Quindi, sono necessarie tecniche per garantire l'integrità del messaggio in grado di rilevare contenuti falsificati o sostituiti (Paragrafo 8.3). Per maggiori dettagli sulla sicurezza WEP si possono consultare [Edney 2003; Wright 2015].

8.8.2 IEEE 802.11i

Subito dopo il rilascio di IEEE 802.11, avvenuto nel 1999 ebbe inizio lo sviluppo di una nuova versione di 802.11 con un meccanismo di sicurezza più efficace. Il nuovo standard 802.11i è stato approvato nel 2004 e, come vedremo, fornisce un solido schema crittografico, un estensibile assortimento di meccanismi di autenticazione e un metodo di distribuzione delle chiavi. Un'eccellente sintesi tecnica di 802.11i è rappresentata da [TechOnline 2012].

Oltre che il client wireless e l'access point (AP), 802.11i definisce un server di autenticazione con il quale l'AP può comunicare (Figura 8.31). Separando il server di autenticazione dall'AP si può fare in modo che sia sufficiente un solo server per

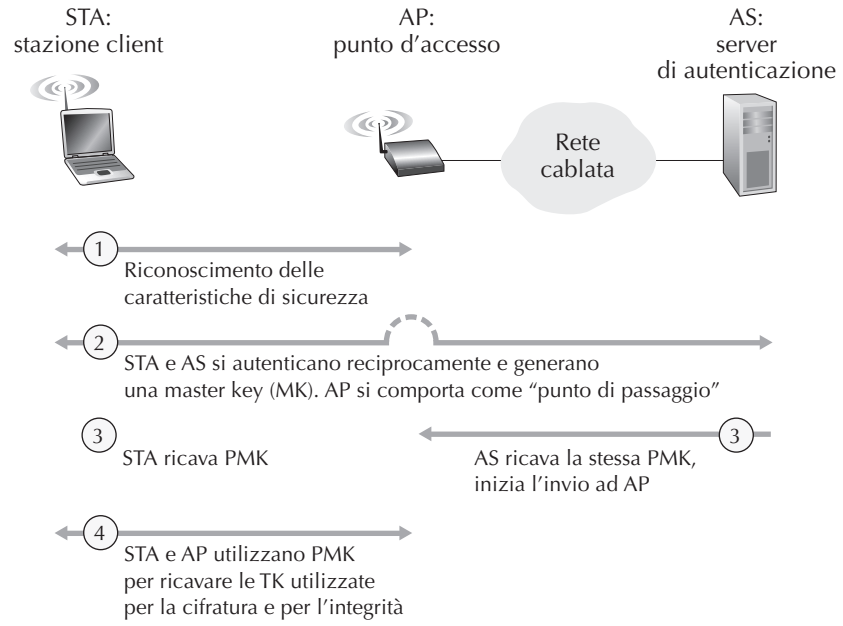

Figura 8.31 802.11i: operazione in quattro fasi.

provvedere a molti punti di accesso, centralizzando così le decisioni di autenticazione e contenendo costi e complessità. Il protocollo opera in quattro fasi.

1. *Riconoscimento*. In questa fase, l'AP si notifica al nodo client wireless e gli comunica le forme di autenticazione e crittografia che può fornire. Il client risponde indicando quali ha scelto. Sebbene sia già in corso uno scambio di messaggi, il client non è stato ancora autenticato e non ha ancora ricevuto una chiave di cifratura. Sono quindi necessari ancora molti passi prima che questo possa comunicare con un host remoto sul canale wireless.
2. *Mutua autenticazione e generazione della chiave principale* (MK, master key). L'autenticazione ha luogo tra il client wireless e il server di autenticazione. In questa fase, l'AP si comporta essenzialmente come un ripetitore inoltrando i messaggi tra il client e il server di autenticazione. Il **protocollo di autenticazione estendibile** (EAP, *extensible authentication protocol*) [RFC 2284] definisce i formati dei messaggi utilizzati nell'interazione tra il client e il server di autenticazione. I messaggi EAP sono incapsulati utilizzando **EAPoL** (EAP over LAN), [IEEE 802.1X], inviati sul collegamento wireless 802.11, e poi decapsulati dall'AP e quindi re-incapsulati secondo il protocollo **RADIUS**, per essere trasmessi su UDP/IP al server di autenticazione (Figura 8.32). Il server e il protocollo **RADIUS** [RFC 2865] non sono previsti dal protocollo, ma sono componenti *de facto* dello standard di 802.11i. È molto probabile che, nell'immediato futuro, RADIUS venga sostituito dal più recente protocollo **DIAMETER** [RFC 3588].

   Con EAP, il server può scegliere una modalità di autenticazione. Dato che 802.11i non predilige un particolare metodo, viene sovente utilizzato lo schema EAP-TLS [RFC 2716] che utilizza tecniche a chiave pubblica (incluso il nonce cifrato e l'hash del messaggio) simili a quelle studiate nel Paragrafo 8.3. Ciò con-

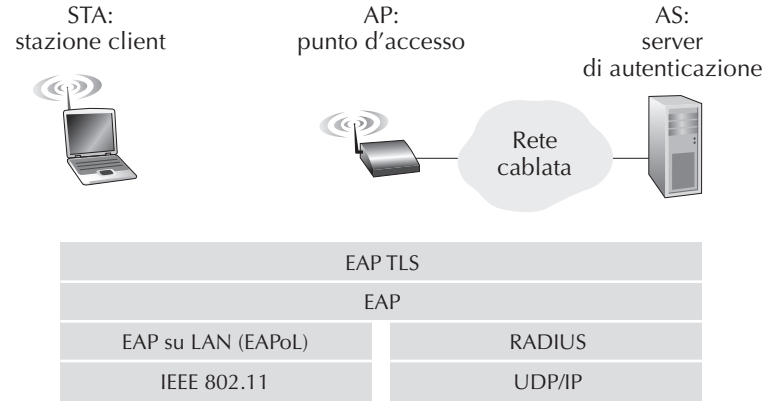

Figura 8.32 EAP è un protocollo punto-a-punto. I messaggi EAP sono incapsulati con EAPoL lungo il collegamento wireless tra il client e l'AP e, con RADIUS su UDP/IP tra l'AP e il server di autenticazione.

sente al client e al server di autenticarsi e di generare una master key nota a entrambe le parti.

3. *Generazione di una master key accoppiata*. La master key è nota solo al client e al server di autenticazione, che la utilizzano per generare una seconda chiave, PMK (master key accoppiata, *pairwise master key*). Il server invia quindi PMK all'AP. Ora client e AP dispongono di una chiave condivisa (in WEP, il problema della distribuzione delle chiavi non era neanche affrontato) ed essendosi reciprocamente autenticati sono quindi completamente operativi.
4. *Generazione della chiave temporale*. Usando PMK, client wireless e AP possono generare chiavi addizionali che saranno utilizzate per la comunicazione. Di particolare interesse è la chiave temporale (TK, *temporal key*), utilizzata a livello di collegamento per la crittografia dei dati inviati lungo il link wireless a un host remoto.

Svariate sono le forme di crittografia fornite da 802.11i, incluso uno schema basato su AES e una versione rinforzata di WEP.

8.9 Sicurezza operativa: firewall e sistemi di rilevamento delle intrusioni

Abbiamo visto nel corso del capitolo che Internet è un ambiente tutt'altro che sicuro, dove i malintenzionati sono in attesa di seminare lo scompiglio. Per un amministratore di rete, il mondo è suddiviso in due schieramenti: i bravi ragazzi e quelli cattivi. La prima schiera include gli appartenenti all'ente che amministra la rete e che dovrebbero poter accedere alle risorse senza particolari restrizioni. Dall'altra parte si trovano i cattivi soggetti, ossia tutti quelli il cui accesso alle risorse di rete deve essere attentamente vagliato. In molte organizzazioni, dai castelli medioevali alle moderne sedi aziendali, esiste uno specifico punto fisico di collegamento con l'esterno dove tutti (o quasi) sono sottoposti a più o meno severi controlli. Un tempo era la guardiola posta all'estremità del ponte levatoio; oggi la *reception* all'ingresso dell'edificio. In una rete di calcolatori, le operazioni – in base alle quali il traffico che entra o esce viene registrato, scartato e/o instradato – avvengono in dispositivi noti come firewall e sistemi di rilevamento (IDS, *intrusion detection system*) e di prevenzione (IPS, *intrusion prevention system*) delle intrusioni.

8.9.1 Firewall

Il **firewall** è una combinazione di hardware e software che separa una rete privata dal resto di Internet e consente all'amministratore di controllare e gestire il flusso di traffico tra il mondo esterno e le risorse interne, stabilendo quali pacchetti lasciare transitare e quali bloccare. Un firewall ha tre obiettivi.

- *Tutto il traffico dall'esterno verso l'interno e viceversa deve passare attraverso il firewall*. La Figura 8.33 mostra un firewall posto proprio sui confini tra la rete am-

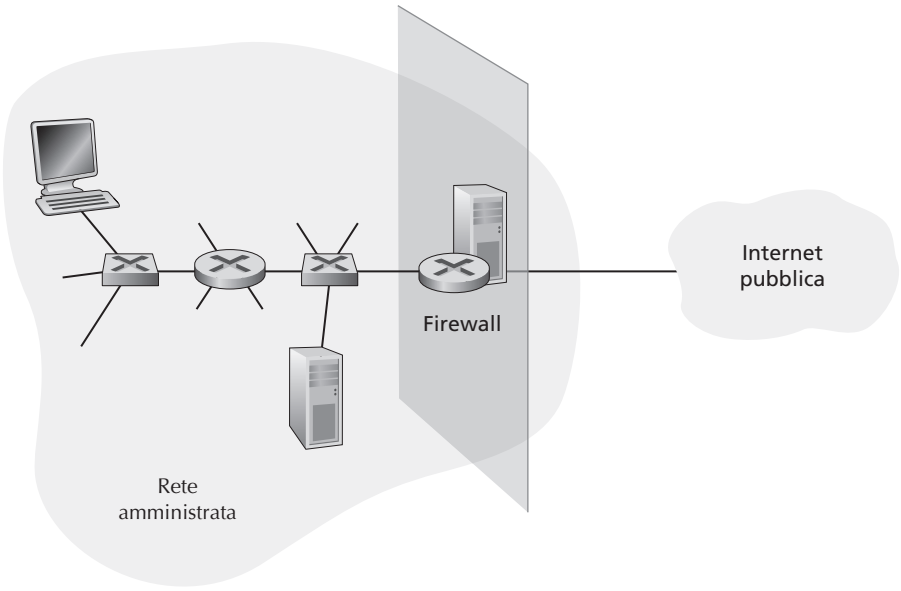

Figura 8.33 Rete amministrata e Internet separate da un firewall.

ministrata e il resto di Internet. Sebbene sia possibile utilizzare livelli multipli di firewall o firewall distribuiti [Skoudis 2006], collocarne uno in un unico punto rende più semplice definirne la politica di accesso.

- *Solo al traffico autorizzato, secondo la definizione nella politica di sicurezza locale, sarà consentito passare*. Dato che tutto il traffico che entra e che esce dalla rete istituzionale passa attraverso il firewall, quest'ultimo può restringere l'accesso al solo traffico autorizzato.
- *Il firewall stesso deve essere immune dalla penetrazione*. Il firewall è un dispositivo collegato alla rete: se non è progettato o installato correttamente, può essere compromesso e, in questo caso, fornisce solo un falso senso di sicurezza, ed è peggio che non averlo del tutto.

Cisco e Check Point sono attualmente tra i due principali fornitori di firewall. Comunque, è facile creare un firewall da una macchina Linux, usando iptable (software di pubblico dominio normalmente rilasciato con Linux).

I firewall possono essere classificati in tre categorie: i tradizionali **filtri di pacchetti** (*packet filter*), i **filtri con memoria di stato** (*stateful filter*) e i **gateway a livello applicativo** (*application level gateway*).

Filtri di pacchetti tradizionali

Di solito una rete privata è collegata al suo ISP (e quindi a Internet) mediante un router attraverso il quale transita tutto il traffico, e che è anche responsabile del **filtraggio dei pacchetti**. Questa operazione prevede in primo luogo l'analisi dell'intestazione dei datagrammi e l'applicazione a questi delle regole di filtraggio stabilite

Tabella 8.5 Politiche e corrispondenti regole di filtraggio per un'organizzazione con indirizzi di rete 130.27/16 e server web con indirizzo IP 130.207.244.203.

Politica	Configurazione del firewall
Nessun accesso web all'esterno	Bloccare tutti i pacchetti uscenti con qualsiasi indirizzo IP di destinazione e porta di destinazione 80
Nessuna connessione TCP entrante, eccetto quelle dirette al solo server web pubblico	Bloccare tutti i pacchetti TCP SYN entranti verso qualsiasi indirizzo IP tranne quelli verso l'indirizzo IP 130.207.244.203 e con porta destinazione 80
Evitare che le radio web consumino tutta la banda disponibile	Bloccare tutti i pacchetti UDP entranti, eccetto i pacchetti DNS
Evitare che la rete possa essere usata per un attacco DoS	Bloccare tutti i pacchetti ICMP ping diretti a un indirizzo broadcast (per esempio 130.207.255.255)
Evitare che la rete possa essere rilevata tramite Traceroute	Bloccare tutti i messaggi ICMP uscenti per TTL scaduto

dall'amministratore di rete per determinare quali devono essere bloccati e quali possono passare.

Le decisioni sono generalmente basate su:

- indirizzo IP sorgente o destinazione;
- tipo di protocollo nel campo del datagramma IP: TCP, UDP, ICMP, OSPF e così via;
- porte sorgente e destinazione TCP o UDP;
- bit di flag TCP: SYN, ACK e così via;
- tipo di messaggio ICMP;
- regole diverse per i datagrammi che escono e entrano dalla rete;
- regole diverse per le diverse interfacce del router.

Un amministratore di rete configura il firewall in base alle politiche dell'organizzazione. Le politiche possono tener presente la produttività degli utenti e l'uso della banda, oltre che le preoccupazioni riguardo alla sicurezza dell'organizzazione. Nella Tabella 8.5 sono elencate alcune politiche che un'organizzazione può stabilire e la loro realizzazione da parte un filtro di pacchetti. Per esempio, se un'organizzazione non vuole alcuna connessione TCP entrante, se non quelle al proprio web server pubblico, può bloccare tutti i segmenti TCP SYN in ingresso, tranne quelli con porta destinazione 80 e indirizzo IP di destinazione corrispondente a quello del web server. Se l'organizzazione non vuole che i propri utenti monopolizzino la banda con applicazioni di radio su Internet, può bloccare tutto il traffico UDP non critico, dato che la radio su Internet spesso viene inviata su UDP. Se un'organizzazione non vuole che si possa dall'esterno creare una mappa della propria rete interna, tramite Traceroute,

può bloccare tutti i messaggi ICMP con TTL esaurito che lasciano la rete dell'organizzazione.

Un'ulteriore procedura di filtraggio prevede la combinazione di numeri di porta e indirizzi, per cui il router può inoltrare tutti i datagrammi Telnet (con porta numero 23) a eccezione di quelli contraddistinti da determinati indirizzi IP. Purtroppo, questa politica non fornisce un'efficace protezione contro i datagrammi che presentano un indirizzo sorgente compreso nella lista di quelli autorizzati, ma che è stato in realtà contraffatto.

Il filtraggio può essere basato sull'impostazione del bit TCP ACK (Paragrafo 3.5) che nel primo segmento delle connessioni TCP è impostato a 0, mentre negli altri ha valore 1. Questo espediente si rivela particolarmente utile quando si vuole che i client interni possano collegarsi a server esterni, evitando però l'operazione inversa. Per fare ciò è sufficiente filtrare tutti i segmenti in arrivo che hanno il bit ACK impostato a 0, in quanto questa procedura "termina" tutte le connessioni TCP che hanno origine all'esterno, ma consente quelle generate internamente.

Le regole del firewall sono implementate nei router tramite liste di controllo degli accessi presenti su ciascuna interfaccia del router. Un esempio di lista di controllo degli accessi per l'organizzazione 222.22/16 è mostrata nella Tabella 8.6 e viene usata sull'interfaccia che collega il router dell'organizzazione a un ISP esterno. Le regole sono applicate a ciascun datagramma che passa attraverso l'interfaccia, dall'alto verso il basso. Le prime due righe consentono agli utenti interni di navigare su Web: la prima regola consente a qualsiasi pacchetto TCP con porta destinazione 80 di lasciare la rete dell'organizzazione, mentre la seconda regola consente a qualsiasi pacchetto TCP con porta sorgente 80 e bit ACK impostato a 1 di entrare nella rete. Si noti che, se una sorgente esterna tentasse di stabilire una connessione TCP con un host interno, la connessione sarebbe bloccata, anche se la porta sorgente o di destinazione è 80. Le seconde due regole, insieme, consentono ai pacchetti DNS di entrare e uscire dalla rete. Riassumendo, questa lista di controllo degli accessi, abbastanza restrittiva, bloc-

Tabella 8.6 Lista di controllo degli accessi per una interfaccia del router.

Azione	Indirizzo sorgente	Indirizzo destinazione	Protocollo	Porta sorgente	Porta destinazione	Bit di flag
consenti	222.22/16	al di fuori di 222.22/16	TCP	>1023	80	qualsiasi
consenti	al di fuori di 222.22/16	222.22/16	TCP	80	>1023	ACK
consenti	222.22/16	al di fuori di 222.22/16	UDP	>1023	53	–
consenti	al di fuori di 222.22/16	222.22/16	UDP	53	>1023	–
blocca	qualunque	qualunque	tutti	qualunque	qualunque	tutti

ca tutto il traffico eccetto il traffico web iniziato dall'interno dell'organizzazione e il traffico DNS. [CERT Filtering 2012] fornisce una lista di regole di filtraggio di pacchetti porta/protocollo consigliati, per evitare parecchie delle note lacune di sicurezza nelle applicazioni di rete esistenti.

Filtri di pacchetti con memoria di stato

In un filtro di pacchetti tradizionale le decisioni di filtraggio vengono prese su ciascun pacchetto in modo indipendente. I filtri con memoria di stato, in realtà, tengono traccia delle connessioni TCP e usano questa conoscenza per prendere le decisioni di filtraggio.

Per comprendere i filtri con memoria di stato, riesaminiamo la lista di controllo degli accessi nella Tabella 8.6. Sebbene sia piuttosto restrittiva, la lista consente a qualsiasi pacchetto che arriva dall'esterno con ACK = 1 e porta sorgente 80 di passare il filtro. Pacchetti di questo tipo potrebbero essere usati dagli attaccanti in tentativi di bloccare i sistemi interni con pacchetti malformati, di svolgere attacchi DoS o di rilevare la struttura della rete interna. La soluzione semplice è quella di bloccare anche i pacchetti TCP ACK, ma in questo modo si negherebbe agli utenti interni dell'organizzazione la possibilità di navigare su Web.

I filtri con memoria di stato risolvono questo problema tenendo traccia di tutte le connessioni TCP in corso in una tabella di connessione. Questo è possibile perché il firewall può osservare l'inizio di una nuova connessione, osservando l'handshake a tre vie (SYN, SYNACK e ACK), e la fine della stessa, quando vede un pacchetto FIN per quella connessione. Il firewall può anche, in modo conservativo, assumere che la connessione è terminata quando non vede alcuna attività su di essa, per esempio per 60 secondi. Un esempio di tabella di connessione di un firewall, mostrata nella Tabella 8.7, indica che, al momento, ci sono tre connessioni TCP in corso, tutte iniziate dall'interno dell'organizzazione. Inoltre, i filtri con memoria di stato includono una nuova colonna "Verifica della connessione" nella loro lista di controllo degli accessi (Tabella 8.8). Si noti che le liste di controllo degli accessi delle Tabelle 8.6 e 8.8 sono identiche, eccetto che ora è indicato che le connessioni dovrebbero essere verificate per due regole.

Consideriamo alcuni esempi per vedere come la tabella di connessione e la lista di controllo degli accessi estesa lavorano insieme. Supponete che un attaccante tenti di far pervenire pacchetti malformati all'interno della rete dell'organizzazione, inviando un datagramma con porta TCP sorgente 80 e flag ACK impostato. Inoltre,

Tabella 8.7 Tabella di connessione per un filtro con memoria dello stato.

Indirizzo sorgente	Indirizzo destinazione	Porta sorgente	Porta destinazione
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

Tabella 8.8 Lista di controllo degli accessi per un filtro con memoria dello stato.

Azione	Indirizzo sorgente	Indirizzo destinazione	Protocollo	Porta sorgente	Porta destinazione	Bit di flag	Verifica della connessione
consenti	222.22/16	al di fuori di 222.22/16	TCP	>1023	80	qual-siasi	
consenti	al di fuori di 222.22/16	222.22/16	TCP	80	>1023	ACK	X
consenti	222.22/16	al di fuori di 222.22/16	UDP	>1023	53	–	
consenti	al di fuori di 222.22/16	222.22/16	UDP	53	>1023	–	X
blocca	tutto	tutto	tutto	tutto	tutto	tutto	

supponete che questo pacchetto abbia porta sorgente 12543 e indirizzo IP sorgente 150.23.23.155. Quando questo pacchetto raggiunge il firewall, questo verifica la lista di controllo degli accessi della Tabella 8.7, che indica che deve essere verificata la tabella di connessione prima di permettere a questo pacchetto di entrare nella rete dell'organizzazione. Il firewall diligentemente verifica la tabella delle connessioni e vede che questo pacchetto non è parte di una connessione TCP in corso e lo blocca. Come secondo esempio, supponete che un utente interno voglia visitare un sito web esterno. Dato che questo utente per prima cosa manda un segmento TCP SYN, la connessione TCP dell'utente viene registrata nella tabella delle connessioni. Quando il web server manda indietro i pacchetti con il bit di ACK necessariamente impostato, il firewall verifica la tabella e vede che una connessione corrispondente è in corso, lascerà così passare questi pacchetti, non interferendo in questo modo con l'attività di navigazione sul Web dell'utente interno.

Gateway a livello applicativo

Da quanto visto finora, abbiamo appreso che il filtraggio dei pacchetti consente di effettuare un controllo, anche se non approfondito, sulle intestazioni IP e TCP/UDP (indirizzi IP, numeri di porta e bit di flag). A questo punto possiamo chiederci che cosa accadrebbe nel caso in cui si volesse fornire un servizio Telnet solo a un gruppo ristretto di utenti interni che dovrebbero autenticarsi prima di iniziare sessioni verso il mondo esterno. Questo compito supera le capacità di un filtro tradizionale o con memoria di stato. Infatti, le informazioni sull'identità degli utenti interni non sono comprese nelle intestazioni IP/TCP/UDP, ma si trovano nei dati a livello di applicazione.

Per ottenere un più elevato livello di sicurezza, i firewall devono combinare il filtraggio dei pacchetti con un **gateway a livello applicativo**: un server specifico attra-

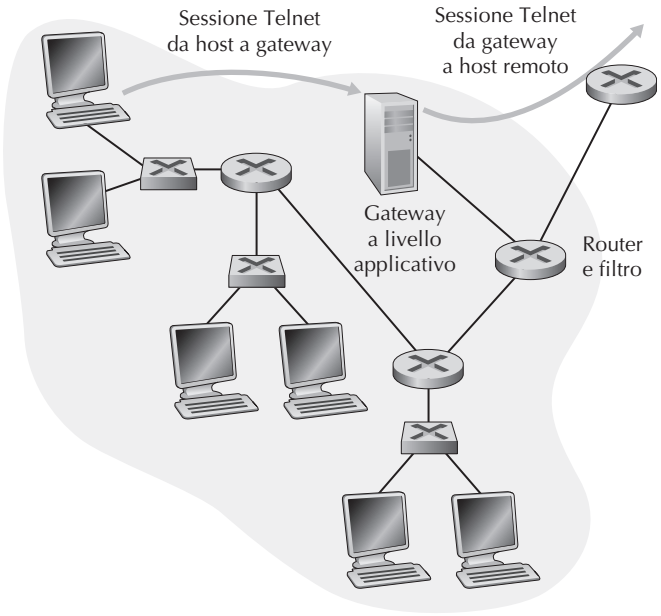

Figura 8.34 Firewall costituito da un gateway applicativo e da un filtro.

verso il quale tutti i dati delle applicazioni (in ingresso e in uscita) sono vincolati a passare. Un **gateway a livello applicativo** guarda oltre le intestazioni IP/TCP/UDP e prende decisioni sulle politiche in base ai dati applicativi. Un host può avere diversi gateway, ma ciascuno costituisce un server separato, con propri processi.

Per meglio comprendere la funzione dei gateway a livello applicativo, immaginiamo di dover progettare un firewall che consenta solo a un limitato gruppo di utenti interni di collegarsi via Telnet con l'esterno e che impedisca l'operazione inversa. Questa politica può essere messa in pratica attraverso l'implementazione di una combinazione di un router per il filtraggio dei pacchetti e di un gateway per l'applicazione Telnet (Figura 8.34). La configurazione del filtro del router blocca tutti i collegamenti eccetto quelli che riportano l'indirizzo IP del gateway. Di conseguenza tutte le connessioni Telnet verso l'esterno devono passare attraverso il gateway. Consideriamo ora un utente che vuole collegarsi in Telnet con l'esterno. Innanzi tutto deve instaurare una sessione Telnet con il gateway e inviargli il proprio identificativo e la password, in modo che il gateway possa controllare se l'utente è accreditato per questo tipo di servizio. In caso negativo, interrompe la connessione, altrimenti (1) chiede all'utente il nome dell'host esterno cui si vuole connettere, (2) imposta una sessione Telnet con il server indicato e (3) inoltra a questo i dati che arrivano dall'utente e viceversa. Pertanto, il gateway non solo concede l'autorizzazione all'utente ma svolge anche la funzione di server e client per Telnet, smistando le informazioni fra l'utente e l'host. Notiamo che, ovviamente, il filtro consente l'esecuzione del passo (2) in quanto la connessione Telnet verso l'esterno è stata avviata dal gateway.

BOX 8.1 **UN CASO DI STUDIO**

Anonimato e privacy

Supponete di voler visitare un sito web controverso (per esempio, il sito di un attivista politico) e che (1) non vogliate rivelare al sito web il vostro indirizzo IP (2) non vogliate che il vostro ISP (a casa o in ufficio) sappia che state visitando tale sito e (3) non vogliate che il vostro ISP veda i dati che vi scambiate col sito. Se usaste l'approccio tradizionale di connettervi direttamente al sito web senza usare alcuna cifratura, non sarebbe soddisfatta alcuna di queste tre richieste. Anche se usaste SSL, non potreste ottenere le prime due: il vostro indirizzo IP sorgente verrebbe mostrato al sito web in ogni datagramma che inviate e l'indirizzo di destinazione di ogni pacchetto potrebbe facilmente essere spiato dal vostro ISP.

Per ottenere anonimità e privacy potreste usare una combinazione di server proxy affidabili e di SSL, come mostrato nella Figura 8.35. Con tale approccio, per prima cosa effettuate una connessione SSL con il proxy affidabile. Quindi inviate sulla connessione SSL la richiesta HTTP di una pagina al sito desiderato. Quando il proxy riceve la richiesta HTTP cifrata con SSL, decodifica la richiesta e inoltra la richiesta HTTP in chiaro al sito web che risponde al proxy, che a sua volta vi inoltra la risposta su SSL. Poiché il sito web vede l'indirizzo IP del proxy e non quello del vostro client, avete ottenuto l'accesso anonimo al sito web. Inoltre, il vostro ISP, poiché tutto il traffico tra voi e il proxy è cifrato, non può invadere la vostra privacy autenticandosi al sito che avete visitato o memorizzando i dati che avete scambiato. Molte aziende (come proxyfy.com) rendono disponibili questi servizi proxy.

Ovviamente in questa soluzione il vostro proxy è a conoscenza di tutto: conosce il vostro indirizzo IP e l'indirizzo IP del sito che state visitando; inoltre vede in chiaro tutto il traffico che vi scambiate. Quindi questa soluzione è tanto buona quanto vi fidate del proxy. Un approccio più robusto, preso dal servizio di anonimizzazione e privacy TOR, consiste nell'instradare il vostro traffico attraverso una serie di proxy che non cooperano tra loro ai danni dell'utente (non collusi) [TOR 2016]. In particolare, TOR permette a proxy individuali indipendenti di aggiungersi al team dei proxy TOR. Quando un utente si connette a un server usando TOR, TOR sceglie casualmente dal proprio insieme di proxy una catena di tre proxy e instrada tutto il traffico tra client e server sulla catena. In questo modo, assumendo che i proxy siano indipendenti, nessuno è a conoscenza che la comunicazione è avvenuta tra il vostro indirizzo IP e il sito web che visitate. Inoltre, sebbene il testo in chiaro sia inviato tra l'ultimo proxy e il server, l'ultimo proxy non sa qual è l'indirizzo IP che sta inviando e ricevendo il testo in chiaro.

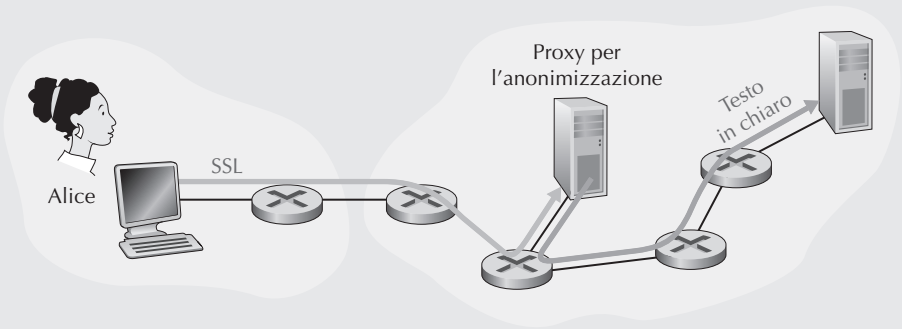

Figura 8.35 Esempio di un proxy che fornisce anonimità e privacy.

Sovente, le reti interne dispongono di vari gateway a livello applicativo: per Telnet, HTTP, FTP e e-mail. In effetti, i server di posta elettronica (Paragrafo 2.3) e i proxy web sono gateway a livello applicativo.

Questa tecnica non è però priva di svantaggi. Innanzitutto occorre un gateway diverso per ciascuna applicazione, inoltre c'è un costo in termini di prestazioni in quanto tutti i dati devono passare attraverso il gateway. Ciò diventa particolarmente problematico quando questo è contemporaneamente utilizzato da vari utenti o applicazioni. Infine, il software del client deve essere in grado di contattare il gateway, quando l'utente fa una richiesta, e comunicargli a quale host esterno si vuole collegare.

8.9.2 Sistemi di rilevamento delle intrusioni

Abbiamo appena visto che un filtro di pacchetti (tradizionale o con memoria di stato) controlla i campi delle intestazioni IP, TCP, UDP e ICMP quando decide quali pacchetti lasciare passare attraverso il firewall. Tuttavia, per rilevare molti tipi di attacchi, abbiamo bisogno di eseguire un **controllo approfondito del pacchetto**, cioè guardare, oltre ai campi dell'intestazione, anche gli effettivi dati applicativi che il pacchetto trasporta. Come abbiamo visto nel Paragrafo 8.9.1, i gateway a livello applicativo fanno spesso un controllo approfondito del pacchetto, ma solo per una specifica applicazione.

Chiaramente, c'è spazio per un altro dispositivo che non solo esamini le intestazioni di tutti i pacchetti che lo attraversano (come un filtro di pacchetti), ma che, diversamente dal filtro, esegua anche un controllo approfondito del pacchetto. Quando un dispositivo di questo tipo osserva un pacchetto o una serie sospetta di pacchetti, può evitare che quei pacchetti entrino nella rete istituzionale. Altrimenti, dato che l'attività è solo ritenuta sospetta, il dispositivo potrebbe lasciare passare i pacchetti, ma inviare avvertimenti all'amministratore di rete, che può osservare il traffico più da vicino e intraprendere le azioni appropriate. Un dispositivo che genera allarmi quando osserva traffico potenzialmente malevolo viene chiamato **sistema di rilevamento delle intrusioni** (IDS, *intrusion detection system*), mentre un dispositivo che filtra il traffico sospetto viene chiamato **sistema di prevenzione delle intrusioni** (IPS, *intrusion prevention system*). In questo paragrafo studieremo entrambi i sistemi ma, poiché l'aspetto tecnico più interessante di questi sistemi è come rilevano il traffico sospetto e non se inviano allarmi o eliminano i pacchetti, d'ora in poi, faremo riferimento ai sistemi IDS e IPS congiuntamente come sistemi IDS.

Un IDS può essere usato per rilevare un'ampia gamma di attacchi, compresi quelli di *mappatura* della rete (tramite l'utilizzo, per esempio, di nmap), la scansione delle porte, la scansione dello stack TCP, gli attacchi DoS di flooding della banda, worm e virus, attacchi di vulnerabilità dei sistemi operativi e quelli di vulnerabilità delle applicazioni. Si veda il Paragrafo 1.6 per una panoramica degli attacchi via rete. Oggi, migliaia di organizzazioni sfruttano sistemi IDS. Molti dei sistemi impiegati sono prodotti da Cisco, Check Point e altri fornitori di apparati di sicurezza, ma molti altri sono sistemi di dominio pubblico, come il popolarissimo sistema IDS Snort, che tratteremo tra breve.

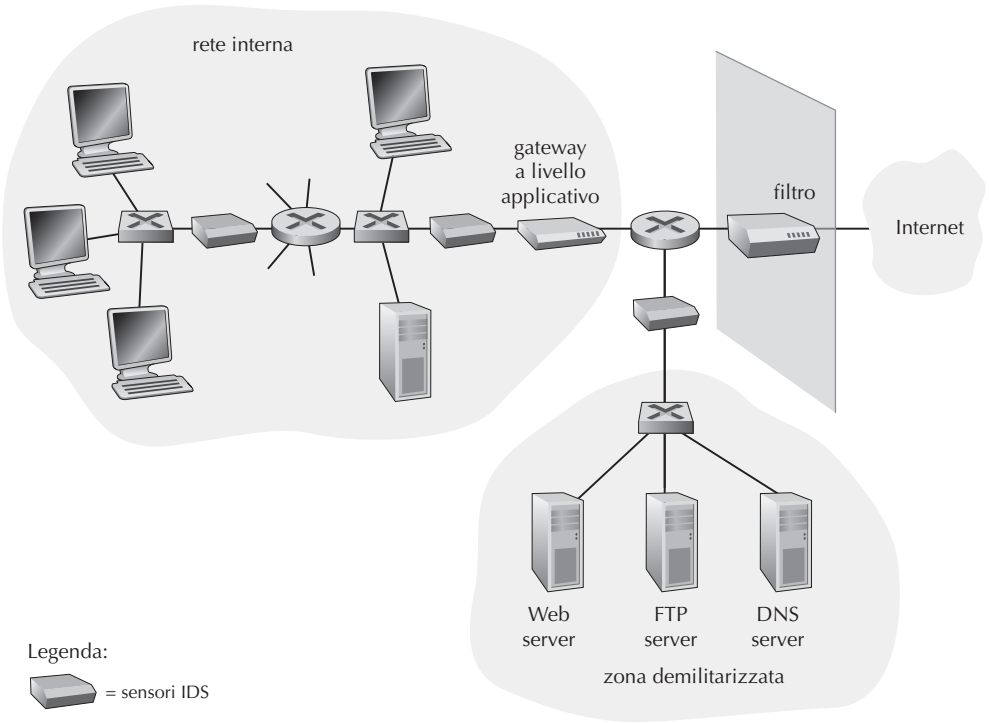

Figura 8.36 Organizzazione che usa un filtro, un gateway applicativo e sensori IDS.

Un'organizzazione può impiegare uno o più sistemi IDS nella propria rete istituzionale (Figura 8.36). Quando vengono impiegati più punti di rilevamento (sensori), questi tipicamente lavorano in maniera coordinata, mandando informazioni sulle attività di traffico sospette a un processore IDS centrale, che raccoglie, integra le informazioni e manda segnalazioni agli amministratori di rete, quando lo ritiene opportuno. Nella Figura 8.36 l'organizzazione ha suddiviso la propria rete in due regioni, una ad alta sicurezza, protetta da un filtro di pacchetti e da un gateway a livello applicativo e monitorata da sensori IDS, e una a bassa sicurezza, chiamata **zona demilitarizzata** (DMZ, *demilitarized zone*), che è protetta solo da un filtro di pacchetti, ma è anch'essa monitorata da sensori IDS. Si noti che la DMZ comprende i server dell'organizzazione che hanno bisogno di comunicare con il mondo esterno, come il web server pubblico e il DNS server autoritativo.

Vi chiederete, a questo punto, perché più sensori IDS piuttosto che posizionarne uno solo dietro il filtro di pacchetti, o anche integrato con esso (Figura 8.36). Vedremo tra breve che un IDS non solo ha bisogno di fare un controllo approfondito dei pacchetti, ma deve anche confrontare ciascun pacchetto che passa con decine di migliaia di "firme" (*signature*): questo può richiedere una significativa quantità di elaborazione, in modo particolare se l'organizzazione riceve Gigabit di traffico al secondo da Internet. Posizionando i sensori IDS più lontano dal flusso di downstream, ciascun

sensore vede solo una frazione del traffico dell'organizzazione e può stare più facilmente al passo con il traffico. Ciò nonostante, i sistemi IDS e IPS sono oggi disponibili con elevate prestazioni, e molte organizzazioni possono effettivamente mettere solo un sensore vicino al proprio router di accesso.

I sistemi IDS sono, in generale, classificati come **sistemi basati sulle firme** o **sistemi basati sulle anomalie**. Un sistema basato su firme mantiene un ampio database di firme degli attacchi. Ciascuna firma è un insieme di regole che riguarda un'attività di intrusione. Una firma può semplicemente essere una lista di caratteristiche di un singolo pacchetto (per esempio numeri di porta sorgente e destinazione, tipo di protocollo e una specifica stringa di bit nel payload) o mettere in relazione una serie di pacchetti. Le firme sono normalmente create da personale qualificato nel campo della sicurezza di rete, che studia gli attacchi noti. Un amministratore di rete dell'organizzazione può personalizzare le firme e aggiungerne di proprie al database.

Operativamente, un IDS basato sulle firme controlla ciascun pacchetto che lo attraversa, confrontandolo con le firme nel proprio database. Se un pacchetto o una serie di pacchetti corrisponde a una firma nel database, l'IDS genera un allarme. L'allarme potrebbe essere inviato all'amministratore di rete in un messaggio di posta elettronica, o potrebbe essere inoltrato a un sistema di gestione di rete, oppure essere semplicemente registrato per venire analizzato in un secondo momento.

I sistemi basati sulle firme, sebbene largamente impiegati, hanno svariate limitazioni. L'aspetto più importante è che, per generare una firma accurata, richiedono una conoscenza pregressa dell'attacco. In altre parole, un IDS basato sulle firme è completamente cieco ai nuovi attacchi che non sono ancora stati registrati. Un altro svantaggio è che, anche se una firma corrisponde, potrebbe non essere il risultato di un attacco, e così si genera un falso allarme. Infine, dato che ogni pacchetto deve essere confrontato con un'ampia raccolta di firme, il sistema può essere sovraccarico e, in realtà, fallire nella rilevazione di molti pacchetti malevoli.

Un IDS basato sulle anomalie crea un profilo di traffico, quando osserva il traffico nelle operazioni normali. Guarda poi i flussi di pacchetti che sono statisticamente insoliti, per esempio una percentuale eccessiva di pacchetti ICMP, una crescita esponenziale nelle scansioni delle porte e nell'attività di ping. L'aspetto più interessate dei sistemi IDS basati sulle anomalie è che non fanno affidamento sulla conoscenza di precedenti aggressioni, per cui possono potenzialmente rilevare nuovi e non documentati attacchi. D'altra parte, è un problema estremamente impegnativo distinguere tra il traffico normale e quello statisticamente insolito. A oggi, la maggior parte degli IDS utilizzati sono quelli basati su firme, sebbene alcuni includano anche alcune funzionalità proprie di quelli basati sulle anomalie.

Snort

Snort è un sistema IDS di pubblico dominio, i cui sorgenti sono disponibili e che conta centinaia di migliaia di installazioni esistenti [Snort 2012; Koziol 2003]. Può funzionare su piattaforme Linux, UNIX e Windows. Usa una generica interfaccia chiamata libpcap per intercettare il traffico, che viene usata anche da Ethereal e da molti

altri software. Può facilmente gestire 100 Mbps di traffico, ma per installazioni con tassi di traffico dell'ordine dei Gbps possono essere necessari più sensori snort.

Per dare uno sguardo a snort, consideriamo un esempio di firma snort:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP PING NMAP"; dsize: 0; itype: 8;)
```

Questa firma corrisponde a qualsiasi pacchetto ICMP che entra nella rete dell'organizzazione (`$HOME_NET`) dall'esterno (`$EXTERNAL_NET`), di tipo 8 (ping ICMP) e ha un payload vuoto (`dsize: 0`). Dato che nmap (Paragrafo 1.6) genera pacchetti ping con queste specifiche caratteristiche, questa firma è progettata per rilevare le scansioni tramite ping di nmap. Quando un pacchetto corrisponde a questa firma, Snort genera un allarme che include il messaggio "`ICMP PING NMAP`".

Forse la cosa più impressionante di Snort è la vasta comunità di utenti ed esperti di sicurezza che mantiene aggiornato il suo database delle firme. Tipicamente, in poche ore dalla scoperta di un nuovo attacco, la comunità Snort scrive e rilascia la firma per quell''attacco, che viene scaricata da centinaia di migliaia di installazioni Snort distribuite nel mondo. Inoltre, usando la sintassi delle firme di Snort, gli amministratori di rete possono personalizzare le firme in base alle necessità della propria organizzazione, sia modificando le firme esistenti, sia creandone di completamente nuove.

8.10 Riepilogo

In questo capitolo abbiamo esaminato i diversi meccanismi che Bob e Alice possono utilizzare per comunicare in modo sicuro. Abbiamo visto che i due sono interessati alla riservatezza (in modo che solo loro siano in grado di comprendere i messaggi che si scambiano), all'autenticazione reciproca (per essere certi che la comunicazione avvenga proprio tra loro) e all'integrità (per avere garanzia che i loro messaggi non siano stati alterati durante il trasferimento). Certamente, la comunicazione sicura non si limita a questi casi e, dal Paragrafo 8.5 al Paragrafo 8.8, abbiamo appreso come la sicurezza può essere utilizzata ai diversi livelli dell'architettura di rete per proteggersi dai malintenzionati che possono sferrare una grande varietà di attacchi.

La prima parte del capitolo ha presentato molti princìpi in merito alla sicurezza delle comunicazioni. Nel Paragrafo 8.2 abbiamo affrontato varie tecniche di crittografia, tra cui quelle a chiave simmetrica e a chiave pubblica. DES e RSA sono stati esaminati come casi per lo studio delle due maggiori classi di tecniche crittografiche attualmente utilizzate nelle reti.

Nel Paragrafo 8.3 abbiamo esaminato due approcci per fornire l'integrità dei messaggi: il codice di autenticazione dei messaggi (MAC) e la firma digitale. I due approcci hanno parecchie analogie: entrambi usano funzioni hash crittografiche ed entrambi ci consentono di verificare la sorgente dei messaggi, oltre che l'integrità del messaggio stesso. Un'importante differenza è che MAC non si affida alla cifratura, mentre la firma digitale richiede un'infrastruttura a chiave pubblica. Entrambe le tecniche sono ampiamente usate nella pratica, come abbiamo visto nei Paragrafi da 8.5 a 8.8. Inoltre, la fir-

ma digitale viene usata per creare i certificati digitali, che sono importanti per verificare la validità delle chiavi pubbliche.

Nel Paragrafo 8.4 abbiamo rivolto la nostra attenzione all'autenticazione e abbiamo sviluppato una serie di protocolli di autenticazione sempre più sofisticati per garantire che uno dei partecipanti alla comunicazione sia effettivamente chi dice di essere e sia attivo. Abbiamo visto che sia la crittografia a chiave pubblica sia quella simmetrica possono giocare un ruolo importante non solo nel celare i dati (cifratura/decifratura) ma anche nell'eseguire l'autenticazione. Nel Paragrafo 8.4 abbiamo esaminato l'autenticazione degli agenti agli estremi di una comunicazione (end-point authentication) e introdotto il nonce in difesa degli attacchi basati su ripetizione.

Nei Paragrafi da 8.5 a 8.8 abbiamo esaminato molti protocolli largamente impiegati per la sicurezza delle reti. Abbiamo visto che la crittografia a chiave simmetrica è il cuore di PGP, SSL, IPSec e della sicurezza wireless, mentre la crittografia a chiave pubblica è cruciale sia in PGP sia in SSL. Abbiamo visto che PGP usa la firma digitale per l'integrità dei messaggi, mentre SSL e IPSec usano MAC. Avendo ora compreso i princìpi base della crittografia e avendo studiato come questi princìpi sono effettivamente usati, siete nella posizione di poter progettare i vostri protocolli di rete sicuri.

Speriamo che Bob e Alice siano studenti di reti e che, supportati delle tecniche apprese nei Paragrafi da 8.2 a 8.4, possano ora comunicare con sicurezza, evitando di essere spiati da Trudy. Ma la riservatezza non è che una piccola parte del quadro della sicurezza in rete.

In misura sempre maggiore, l'attenzione nella sicurezza di rete è focalizzata sulle infrastrutture di rete che proteggono da possibili assalti di malintenzionati. Nell'ultima parte del capitolo abbiamo esaminato i firewall e i sistemi IDS che controllano i pacchetti che entrano ed escono dalla rete di una organizzazione.

In questo capitolo abbiamo percorso molto strada mentre ci siamo concentrarti sugli aspetti più importanti della moderna sicurezza delle reti. I lettori che desiderano approfondire questi aspetti sono incoraggiati a considerare i riferimenti citati in questo capitolo. In particolare raccomandiamo: [Skoudis 2006] per gli attacchi e la sicurezza operativa; [Kaufman 1995] per la crittografia e per come si applica alla sicurezza di rete; [Rescorla 2001] per un'approfondita ma interessante trattazione su SSL; e [Edney 2003] per una completa discussione della sicurezza di 802.11, compreso uno studio di WEP e delle sue lacune.

Domande e problemi

Domande di revisione

PARAGRAFO 8.1

R1. Quali sono le differenze fra riservatezza e integrità di un messaggio? Si può avere l'una senza l'altra? Argomentate la risposta.