

Università di Genova
Facoltà di Ingegneria

12. Sicurezza nelle reti di telecomunicazioni

Prof. Raffaele Bolla



Sicurezza nelle reti

- Ci sono tre aree in cui bisogna intervenire per rendere una rete sicura
 - **Riservatezza**: il messaggio deve essere accessibile (visualizzabile o rilevabile la sua presenza) solo ad entità autorizzate.
 - **Autenticazione**: L'identità delle entità coinvolte nella comunicazioni deve poter essere verificata.
 - **Integrità** (ed eventuale "firma"): impedire che i dati possano essere modificati se non da autorità autorizzate (con firma: anche legate all'autore, che non ne possa disconoscere la paternità).

6.2

Sicurezza nelle reti Attacchi

Passivi

- **Accesso al contenuto:** venire a conoscenza di informazioni riservate. Ad esempio lo *Sniff* (il fiutare) di pacchetti su LAN a mezzo condiviso.
- **Analisi del traffico:** senza vedere i contenuti specifici, riconoscere l'entità dei comunicanti e tipo e frequenza dei messaggi.
- Sono difficili da rilevare, quindi si devono prevenire.

6.3

Sicurezza nelle reti Attacchi al contenuto - Esempio di sniffing

```

1 0.000000 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [SYN] Seq=2175924618 Ack=0 Win=5840 Len=0
2 0.047915 wpop10.libero.it abete.reti.dist.unige TCP http > 32811 [SYN, ACK] Seq=395931056 Ack=2175924619 Win=24616 Len=0
3 0.047968 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175924619 Ack=395931056 Win=5840 Len=0
4 0.048136 abete.reti.dist.unige wpop10.libero.it HTTP POST /enail.php HTTP/1.1
5 0.080256 wpop10.libero.it abete.reti.dist.unige TCP http > 32811 [ACK] Seq=395931056 Ack=2175925157 Win=24616 Len=0
6 0.080281 abete.reti.dist.unige wpop10.libero.it HTTP Continuation
7 0.223209 wpop10.libero.it abete.reti.dist.unige TCP http > 32811 [ACK] Seq=395931056 Ack=2175925329 Win=24616 Len=0
8 5.503723 wpop10.libero.it abete.reti.dist.unige HTTP HTTP/1.1 200 OK
9 5.503763 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175925329 Ack=395931407 Win=6432 Len=0
10 5.575033 abete.reti.dist.unige wpop10.libero.it HTTP GET /error.html HTTP/1.1
11 5.595797 wpop10.libero.it abete.reti.dist.unige HTTP HTTP/1.1 200 OK
12 5.595843 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175925881 Ack=395932855 Win=8688 Len=0
13 5.596326 wpop10.libero.it abete.reti.dist.unige HTTP Continuation
14 5.596367 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175925881 Ack=395934303 Win=11984 Len=0
15 5.596633 wpop10.libero.it abete.reti.dist.unige HTTP Continuation
16 5.598706 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175925881 Ack=395934857 Win=14480 Len=0
17 5.638893 wpop10.libero.it abete.reti.dist.unige HTTP GET /old/xaw_rc/template_xaw/images/spacer.gif HTTP/1.1
18 5.643663 abete.reti.dist.unige wpop10.libero.it TCP 32812 > http [SYN] Seq=2171723094 Ack=0 Win=5840 Len=0
19 5.667860 wpop10.libero.it abete.reti.dist.unige HTTP HTTP/1.1 200 OK
20 5.667898 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175926465 Ack=395935211 Win=17376 Len=0
21 5.668389 abete.reti.dist.unige wpop10.libero.it HTTP GET /old/xaw_rc/template_xaw/images/button_err_back.gif HTTP/1.1
22 5.677590 wpop10.libero.it abete.reti.dist.unige TCP http > 32812 [SYN, ACK] Seq=1219537146 Ack=2171723095 Win=24616 Len=0
23 5.677615 abete.reti.dist.unige wpop10.libero.it TCP 32812 > http [ACK] Seq=2171723095 Ack=1219537147 Win=5840 Len=0
24 5.677745 abete.reti.dist.unige wpop10.libero.it HTTP GET /old/xaw_rc/template_xaw/images/header_err.gif HTTP/1.1
25 5.699033 wpop10.libero.it abete.reti.dist.unige HTTP HTTP/1.1 200 OK
26 5.707494 wpop10.libero.it abete.reti.dist.unige TCP http > 32812 [ACK] Seq=1219537147 Ack=2171723683 Win=24028 Len=0
27 5.741745 abete.reti.dist.unige wpop10.libero.it TCP 32811 > http [ACK] Seq=2175927058 Ack=395935708 Win=17376 Len=0
28 9.096882 wpop10.libero.it abete.reti.dist.unige HTTP HTTP/1.1 200 OK
29 9.096723 abete.reti.dist.unige wpop10.libero.it TCP 32812 > http [ACK] Seq=2171723683 Ack=1219538216 Win=7483 Len=0

```

6.4

Sicurezza nelle reti Attacchi al contenuto - Esempio di sniffing

```

Frame 6 (238 on wire, 238 captured)
  Arrival Time: Dec 10, 2002 13:39:46.422800000
  Time delta from previous packet: 0.000025000 seconds
  Time relative to first packet: 0.080281000 seconds
  Frame Number: 6
  Packet Length: 238 bytes
  Capture Length: 238 bytes
Ethernet II
  Destination: 00:00:0c:03:de:0a (Cisco_03:de:0a)
  Source: 00:e0:18:a0:36:cc (Asustek_a0:36:cc)
  Type: IP (0x0800)
Internet Protocol, Src Addr: abete.reti.dist.unige.it (130.251.8.11), Dest Addr: wpp10.libero.it (193.70.192.46)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00; Default; ECN: 0x00)
  Total Length: 224
  Identification: 0x2efe
  Flags: 0x04
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (0x06)
  Header checksum: 0xfe9e (correct)
  Source: abete.reti.dist.unige.it (130.251.8.11)
  Destination: wpp10.libero.it (193.70.192.46)
Transmission Control Protocol, Src Port: 32811 (32811), Dest Port: http (80), Seq: 2175925157, Ack: 396931056
  Source port: 32811 (32811)
  Destination port: http (80)
  Sequence number: 2175925157
  Next sequence number: 2175925329
  Acknowledgment number: 396931056
  Header length: 32 bytes
  Flags: 0x018 (PSH, ACK)
  Window size: 5840
  Checksum: 0x24ab (correct)
  Options: (12 bytes)
  Hypertext Transfer Protocol
    Content-Type: application/x-www-form-urlencoded\r\n
    Content-Length: 100\r\n
    \r\n
    Data (100 bytes)
  
```

6.5

Sicurezza nelle reti Attacchi al contenuto - Esempio di sniffing

```

0000 00 00 0c 03 de 0a 00 e0 18 a0 36 cc 08 00 45 00 ....P..à . 6i..E.
0010 00 e0 2e fe 40 00 40 06 fe 9e 82 fb 08 0b c1 46 .à.p@.ê. p.,.ù..ÁF
0020 c0 2e 80 2b 00 50 81 b1 fb a5 17 99 6d b0 80 18 À.+.P.± 0%..m°..
0030 16 d0 24 ab 00 00 01 01 08 0a 00 0e d7 79 22 87 .0$«. ... ..x.y".
0040 96 9d 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 ..Content-Type:
0050 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 77 77 applicat ion/x-ww
0060 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64 65 w-form-u rlencode
0070 64 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 d..Conte nt-Lengt
0080 68 3a 20 31 30 30 0d 0a 0d 0a 54 6f 6d 69 6e 69 h: 100.. ..lomi ni
0090 5f 3d 6c 69 62 65 72 6f 2e 69 74 26 4c 4f 47 49 p=libero .it&LOGI
00a0 4e 3d 75 74 65 6e 74 65 26 50 41 53 53 57 44 3d N=utente &PASSWD=
00b0 70 61 73 73 26 63 68 6f 69 63 65 3d 6c 69 62 65 pass&cho ice=libe
00c0 72 6f 26 41 63 74 5f 4c 6f 67 69 6e 2e 78 3d 35 ro&Act_L ugin,x=5
00d0 26 41 63 74 5f 4c 6f 67 69 6e 2e 79 3d 39 26 &Act_Log in,y=9&A
00e0 53 74 5f 4c 6f 67 69 6e 3d 45 6e 74 72 61 ct_Login =Entra
  
```

Login: utente
Password: pass

6.6

Sicurezza nelle reti

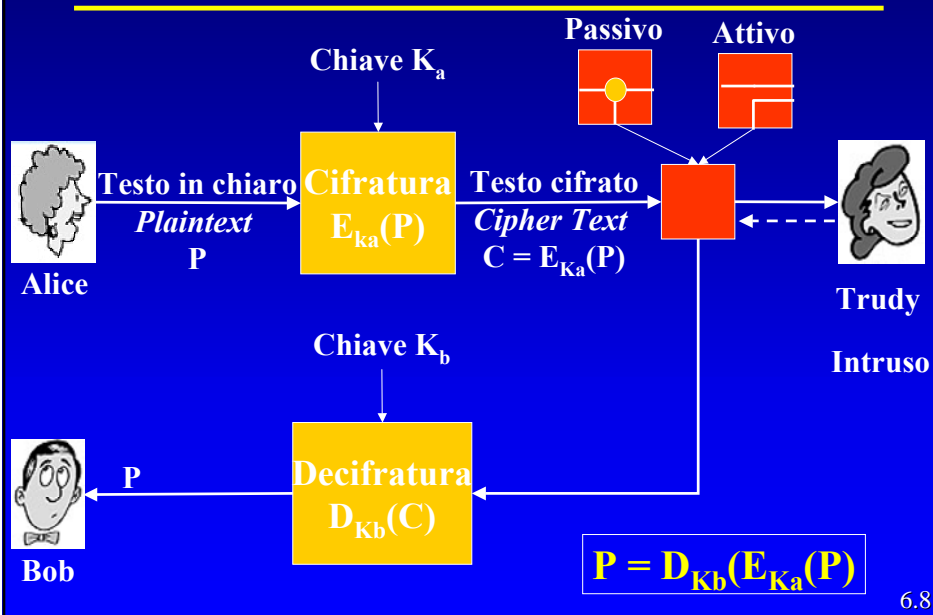
Attacchi

Attivi

- **Sostituzione:** Farsi passare per un altro
Ad esempio lo *Spoofing* (imbroglio) IP.
- **Replica:** copiare e riproporre un messaggio per ottenere effetti non autorizzati (ad esempio, un doppio versamento).
- **Alterazione:** modifica anche solo dell'ordine
- **Negazione del servizio:** inibire l'uso o la gestione di un sistema (anche dell'intera rete), ad esempio per impedire la generazione o arrivo di messaggi di allarme (*SYN Attack*).
- Possono sia essere rilevati e quindi fermati che prevenuti

6.7

Riservatezza: Cifratura



6.8

Attacchi al testo cifrato

- Attacco al testo cifrato (*chipertext only*): chi attacca ha a disposizione solo la conoscenza di una certa quantità di testo cifrato.
- Attacco al testo in chiaro conosciuto (*known plaintext*): chi attacca conosce alcuni campioni di testo in chiaro e i corrispondenti messaggi cifrati.
- Attacco al testo in chiaro scelto (*chosen plaintext*): chi attacca ha la possibilità di criptare il testo in chiaro desiderato.

6.9

Attacchi al contenuto

- Per scardinare un algoritmo di cifratura esistono due tecniche:
 - **Criptoanalisi**: che si basa sulla natura degli algoritmi, su campioni, su caratteristiche statistiche di P.
 - **Forza bruta**.

Dim. chiave	# di chiavi possibili	Tempo (1 cifr./ μ s)	Tempo (10^6 cifr./ μ s)
32	$2^{32} = 4,3 \times 10^9$	35,8 min.	2,15 ms
56	$2^{56} = 7,2 \times 10^{16}$	1142 anni	10,01 ore
128	$2^{128} = 3,4 \times 10^{38}$	$5,4 \cdot 10^{24}$ anni	$5,4 \cdot 10^{18}$ anni
168	$2^{168} = 3,7 \times 10^{50}$	$5,9 \cdot 10^{36}$ anni	$5,9 \cdot 10^{30}$ anni

6.10

Algoritmi di sostituzione

- È una tecnica antica (Giulio Cesare);
- ogni lettera o gruppo di lettere è sostituito da un'altra lettera o gruppo di lettere
 - un semplice algoritmo consiste con la sostituzione di ogni lettera con la terza che segue in ordine alfabetico;
 - generalizzando, si può pensare di sostituire ogni lettera con la *k-esima* successiva, *k* rappresenta in questo caso la chiave dell'algoritmo;
 - un miglioramento consiste nel mappare ogni lettera in un'altra (sostituzione monoalfabetica):

testo in chiaro:	a b c d e f g h i j k l m n o p q r s t u v x y w z
testo cifrato:	Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

- in questo caso le possibili chiavi sono $26! \approx 4 \times 10^{26}$ (provando 1 chiave ogni nanosecondo ci vorrebbero 10^{10} anni per provarle tutte).

6.11


Algoritmi di sostituzione

- Possono essere scardinati con la criptoanalisi:
 - conoscendo una piccola parte di testo cifrato,
 - nota la frequenza statistica delle singole lettere o gruppi di esse,
 - conoscendo alcune parole che probabilmente sono presenti nel testo.

6.12

Algoritmi di trasposizione

- Gli algoritmi di sostituzione scambiano i simboli ma ne mantengono l'ordine.
- Gli algoritmi di trasposizione, al contrario, mantengono i simboli ma ne modificano l'ordine.

Chiave  →

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Testo in chiaro:
pleasetransferonemilliondollarstomysw
issbankaccountsixtwo

Testo cifrato:
AFLLSKSOSELAWAIATOSSCTC
LNMOMANTESILYNTWRNNTSO
WDPAEDOBUEIRICXB

6.13

Algoritmi di trasposizione

- Per scardinarli è necessario sapere che si tratta di una trasposizione
 - si può verificare controllando la frequenza delle diverse lettere.
- Si ipotizza il numero di colonne
 - meglio se si conoscono possibili parole presenti;
 - si devono riordinare le colonne;
 - si possono usare informazioni statistiche su blocchi di più lettere.

6.14

Un algoritmo sicuro

- Si prenda una sequenza di bit casuale, lunga quanto il testo da cifrare.
- Il testo cifrato viene ricavato dall'operazione di XOR tra la chiave e il testo.
- Questo algoritmo è immune da attacchi:
 - la teoria dell'informazione afferma che non c'è più nessuna informazione nel messaggio cifrato, in quanto tutti i testi in chiaro hanno la stessa probabilità di corrispondere a quel particolare messaggio.
- Ci sono evidenti svantaggi:
 - la chiave non può essere memorizzata;
 - la chiave può essere molto lunga;
 - la lunghezza massima del testo codificato è limitata dalla lunghezza della chiave.

6.15

Cifratura a chiave simmetrica

- $K_A = K_B = K$: una sola chiave
- Tradizionalmente si basava su algoritmi semplici (come la sostituzione e la trasposizione).
- Oggi la tendenza è di utilizzare algoritmi talmente complessi la cui conoscenza è inutilizzabile per la decodifica senza la chiave.
- Deve rispettare due requisiti per essere sicura:
 - Robustezza dell'algoritmo: anche conoscendo l'algoritmo ed avendo campioni di testo in chiaro e cifrato, l'intruso non deve essere in grado di decifrare il testo e scoprire la chiave
 - Mittente e destinatario devono poter ottenere in modo sicuro la chiave e custodirla efficacemente.

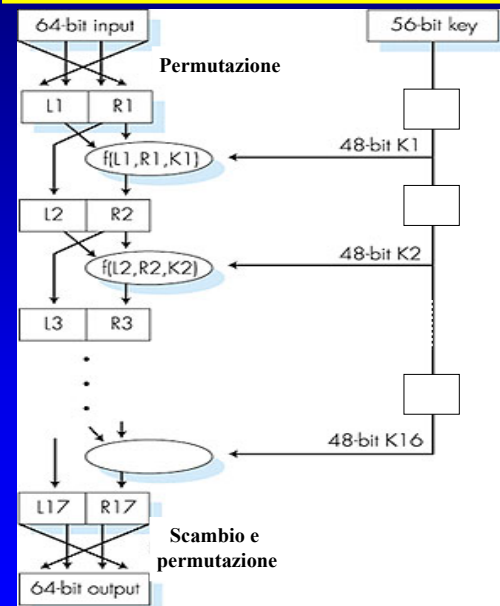
6.16

Cifratura a chiave simmetrica Data Encryption Standard (DES)

- Nasce nel 1977 e viene aggiornato nel 1993,
- E' stato adottato dal U. S. *National Bureau of Standard* (oggi *National Institute for Standard and Technology*, NIST)
- L' algoritmo vero e proprio si chiama *Data Encryption Algorithm* (DEA):
 - Opera su blocchi da 64 bit.
 - Usa una chiave da 56 bit.
 - Si compone di 19 stadi:
 - » Una prima permutazione
 - » 16 stadi parametrizzati da una variante della chiave K_i , $i=1,\dots,16$
 - » Uno scambio dei 32 bit destri con i sinistri
 - » Una permutazione inversa alla prima

6.17

Cifratura a chiave simmetrica Data Encryption Standard (DES)



- In genere viene usato in unione con un concatenamento (CBC).
- La decifratura avviene con lo stesso meccanismo ma usando le chiavi in ordine inverso.
- La complessità dell'algoritmo risiede nella funzione $f(\cdot)$.

6.18

Cifratura a chiave simmetrica Data Encryption Standard (DES)

- Per quanto concerne la robustezza, sono stati indetti tre concorsi (*challenger*) per violarlo:
 - *Challenger I* (1997): Scardinato in 4 mesi;
 - *Challenger II* (1998): Scardinato in 56 ore
 - *Challenger II* (1999) scardinato in 22 ore e 15 min. (testate 245×10^9 chiavi al sec.)
- Ad oggi, (nella sua forma con chiave a 56 bit) non è considerato molto sicuro.

6.19

Cifratura a chiave simmetrica Triplo-DEA (T-DEA)

- Standardizzato dall'ANSI (1985) come X 9.17 e parte del DES dal 1999
- Usa 3 chiavi da 56 bit: K_1 , K_2 , K_3 .
- Opera come segue:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$
- Questo significa che ha una chiave di lunghezza complessiva pari a 168 bit
- Si può cifrare e decifrare il DEA ponendo tutte le chiavi uguali (pensato per avere compatibilità con il DEA originale).
- Si può usare una chiave da 112 bit ponendo $K_1 = K_3$.

6.20

Cifratura a chiave simmetrica AES

- Il *National Institute of Standards and Technology* (NITS), ente governativo americano, nel 1997 ha lanciato un concorso pubblico per individuare un nuovo standard di crittografia, per uso generico del governo americano
 - il nome dello standard sarebbe stato AES (*Advanced Encryption Standard*);
 - lo scopo del concorso pubblico era quello di evitare ogni possibile sospetto sul nuovo standard.
- I requisiti richiesti erano:
 - utilizzo di crittografia a chiave simmetrica;
 - progetto completamente pubblico;
 - chiavi di lunghezza 128, 192 e 256 bit;
 - implementazione hw e sw;
 - algoritmo doveva essere liberamente utilizzato o non avere restrizioni particolari.

6.21

Cifratura a chiave simmetrica AES - Rijndael

- L'algoritmo prescelto è stato il Rijndael (Rijmen e Daemen)
 - la selezione si è basata su criteri di sicurezza, efficienza, semplicità, flessibilità e requisiti di memoria (per sistemi embedded).
- Supporto chiavi e blocchi di cifratura da 128 a 256 bit, a passi di 32 bit
 - blocchi e chiavi possono avere diversa lunghezza.
- L'utilizzo più frequente prevede chiavi di 128 o 256 bit e blocchi di 128 bit.

6.22

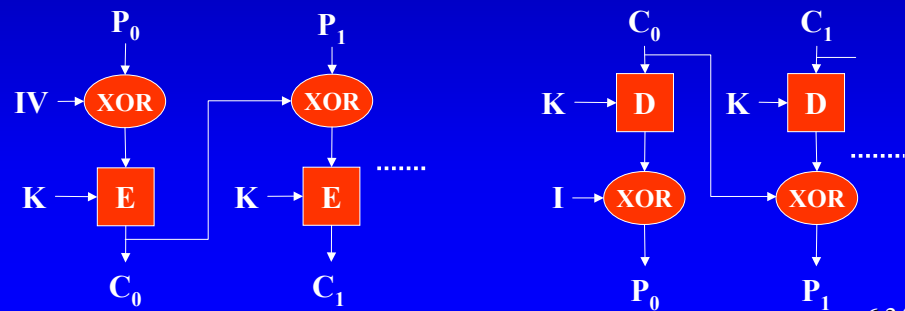
Cifratura a chiave simmetrica

- Nonostante la complessità, il DES e l'AES consistono in una sostituzione monoalfabetica
 - lo stesso testo in ingresso genera sempre gli stessi blocchi in uscita;
 - ci sono problemi di sicurezza simili a quelli già visti per gli algoritmi di sostituzione più semplici.
- Per risolvere il problema è necessario introdurre una ulteriore aleatorietà nel processo
 - varie metodologie sono presenti per concatenare blocchi successivi tra di loro.

6.23

Chiper Block Chaining

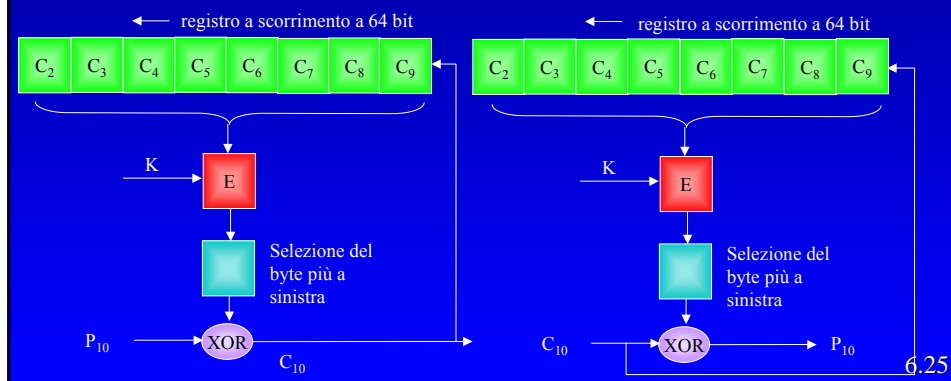
- Ogni blocco di testo in chiaro è posto in XOR con il blocco cifrato precedente prima di essere codificato.
- Il primo blocco è posto in XOR con un vettore di inizializzazione IV (*Initial Vector*).



6.24

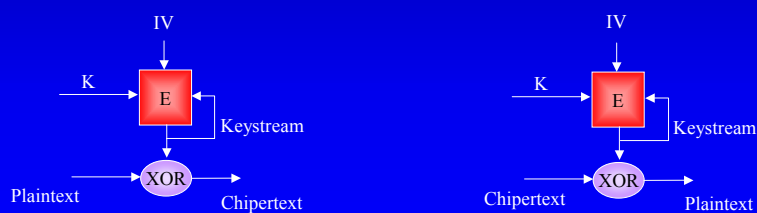
Chiper Feedback

- Con questo algoritmo è possibile codificare un byte alla volta.
- Un errore su un byte sulla linea condiziona un intero blocco.



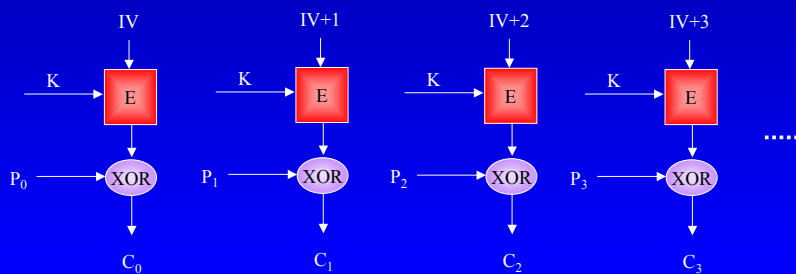
Stream Cipher

- Cifra un vettore di inizializzazione la prima volta, poi cifra ciclicamente l'uscita della codifica precedente.
- 1 bit modificato nella stringa cifrata genera 1 solo bit di errore nel messaggio originale.
- Soggetto al *keystream reuse attack* se non si cambia K o IV :
 - XOR di 2 cipertext genera l'XOR dei due plain text;
 - è possibile ricavare il testo originale con una analisi statistica.



Counter Mode

- Utilizzato per poter decifrare singoli blocchi di un ciphertext senza dover decifrare tutti i blocchi precedenti.
- Se non si cambiano K e IV ad ogni codifica presenta le stesse debolezze dell'algoritmo precedente.



6.27

Cifratura a chiave simmetrica

- Collocazione dei dispositivi di cifratura, due possibilità:
 - Sulle linee (il pacchetto rimane vulnerabile nei commutatori)
 - Sui dispositivi terminali (non è possibile cifrare anche le intestazioni ma solo i dati)
- L'ottimo è utilizzare ambedue i metodi.

6.28

Cifratura a chiave pubblica

- Utilizza due chiavi:
 - Una chiave K_A usata per la cifratura che viene resa pubblica (chiave pubblica).
 - Una chiave K_B usata per la decifratura che viene mantenuta segreta (chiave privata).
- Si evita (ma solo parzialmente!) il problema della distribuzione della chiave.
- Deve avere tre requisiti
 - $D_{K_B}(E_{K_A}(P)) = P$
 - Non deve essere possibile dedurre K_B da K_A .
 - K_B non deve poter essere dedotta tramite cifratura di testi noti

6.29

Cifratura a chiave pubblica Rivest, Shamir, Adelson (RSA)

Scelta delle chiavi

- Si scelga due numeri primi grandi (ad esempio da 1024 bit): p e q .
- Si calcoli $n = p \cdot q$, $z = (p - 1)(q - 1)$.
- Si scelga e (con $e < n$) tale che non abbia fattori comuni con z (e e z sono "primi relativi").
- Si scelga d tale che $ed - 1$ sia esattamente divisibile per z (in altre parole $e \cdot d \bmod z = 1$).
- La chiave pubblica $K_A = (n, e)$ e la chiave privata $K_B = (n, d)$.

6.30

Cifratura a chiave pubblica Rivest, Shamir, Adelson (RSA)

- Dati (n, e) e (n, d) :
 - Per cifrare una sequenza di bit m , si calcola:
 $c = m^e \bmod n$ (ossia il resto di m^e diviso n)
 - Per decifrare una sequenza di bit c ricevuta, si calcola:
 $m = c^d \bmod n$ (ossia il resto di c^d diviso n)
- Ciò che accade è che

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

6.31

Cifratura a chiave pubblica Rivest, Shamir, Adelson (RSA)

Bob sceglie $p = 5, q = 7$.

Quindi $n = 35, z = 24$.

$e = 5$ (così e, z sono primi relativi).

$d = 29$ (così $ed-1$ è divisibile esattamente per z).

Cifra:	<u>Lettera</u>	<u>m</u>	<u>m^e</u>	<u>$c = m^e \bmod n$</u>
		12	248832	17
Decifra:	<u>c</u>	<u>c^d</u>	<u>$m = c^d \bmod n$</u>	<u>Lettera</u>
	17	481968572106750915091411825223072000	12	

6.32

Cifratura a chiave pubblica Rivest, Shamir, Adelson (RSA)

- Perché vale $m = (m^e \bmod n)^d \bmod n$?
- La base è un risultato della teoria dei numeri, ossia se p e q sono primi e $n = p \cdot q$ allora:

$$x \bmod n = x^{e \bmod (p-1)(q-1)} \bmod n$$
- $(m^e \bmod n)^d \bmod n = m^{ed} \bmod n =$
 $= m^{ed \bmod (p-1)(q-1)} \bmod n =$
 (grazie al risultato della teoria dei numeri di cui sopra)
 $= m^1 \bmod n =$
 (dato che si è scelto ed divisibile per $(p-1)(q-1)$ con resto 1)
 $= m$

6.33

Cifratura a chiave pubblica Rivest, Shamir, Adelson (RSA)

- Si osservi che l' algoritmo funziona anche a chiavi invertite.
- Il meccanismo è sicuro perché, al momento, non sono noti algoritmi veloci per la fattorizzazione dei numeri (altrimenti basterebbe fattorizzare n)
- Il problema della cifratura a chiave pubblica è il tempo di elaborazione, rispetto alla chiave simmetrica:
 - In software è 100 volte più lenta
 - In hardware è da 1000 a 10.000 volte più lenta
- Allora viene usato, in genere, solo per lo scambio di una chiave simmetrica di sessione.

6.34

Integrità e firma elettronica

- La firma elettronica è la forma più completa di verifica di integrità. Tale tipo di firma dovrebbe far sì che:
 - L'integrità del messaggio originale sia assicurata.
 - La firma sia legata indissolubilmente al messaggio.
 - La firma sia verificabile (permette di identificare chi ha firmato).
 - La firma sia non falsificabile e non rifiutabile (solo quell'individuo deve poter fare quella firma e non deve poterla disconoscere).

6.35

Firma elettronica

- Un modo per firmare il proprio documento è quello di codificarlo con la propria chiave privata.
- Dato che solo il proprietario ha la chiave privata, questo assicura che solo lui può averlo codificato, e chiunque può verificare che è stato lui a codificarlo usando la sua chiave pubblica e ritrovando il messaggio.
- Questo procedimento ha un limite:
 - La cifratura di un messaggio (con chiave pubblica) è una operazione onerosa se fatta su grandi quantità di dati. E lo stesso vale per la decifratura, obbligatoria per poter leggere il messaggio

6.36

Firma elettronica

- Un meccanismo alternativo che impone un minor onere computazionale è quello del *message digest* (sunto del messaggio).
- Il principio è simile a quello dei codici a rivelazione d'errore, si applica ad un messaggio p una funzione $H()$ il cui risultato è un blocco di dati d_p (il *digest*) con dimensioni molto minori di p . Tale *digest* deve essere legato in modo univoco la messaggio originale
- Tale funzione $H()$ viene chiamata funzione di **hash**.

6.37

Integrità e Firma elettronica

Digest

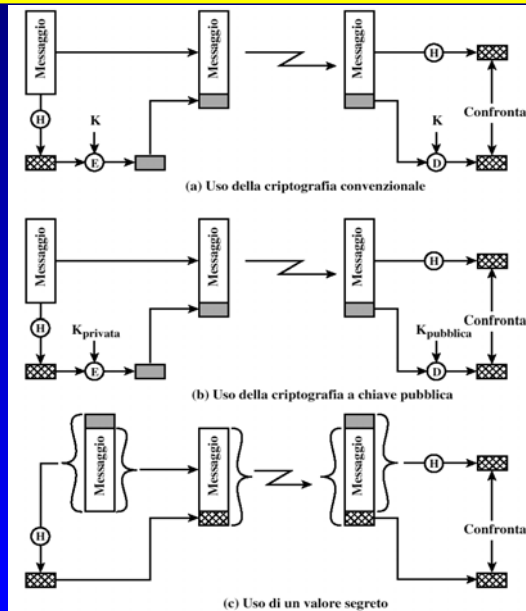
- La funzione di *hash* $H()$ deve avere le seguenti proprietà:
 - Deve poter essere applicata a messaggi di qualunque dimensione.
 - Deve produrre un risultato di lunghezza fissa.
 - Deve essere relativamente semplice e veloce da calcolare.
 - Per ogni *digest* d dato, deve essere computazionalmente impossibile trovare x tale che $H(x) = d$ (non invertibilità).
 - Per ogni messaggio x deve essere computazionalmente impossibile trovare $y \neq x$ tale che $H(y) = H(x)$ (impedisce falsificazioni).
 - Deve essere computazionalmente impossibile trovare una qualsiasi coppia (x, y) tale $H(x) = H(y)$.

6.38

Integrità e Firma elettronica

Digest

Possibili usi del
digest per la
verifica
dell'integrità



Integrità e Firma elettronica

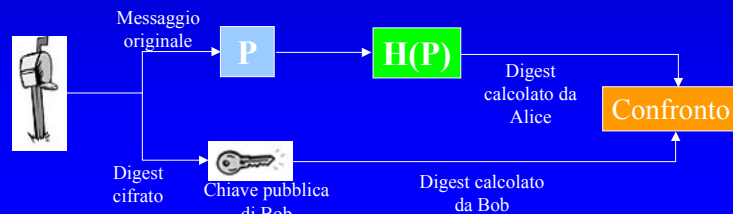
Digest

- Si può usare il *digest* cifrato con la chiave privata; corrisponde a firmare il messaggio

Bob firma il messaggio e lo invia



Alice riceve il messaggio e verifica la firma



6.40

Integrità e Firma elettronica

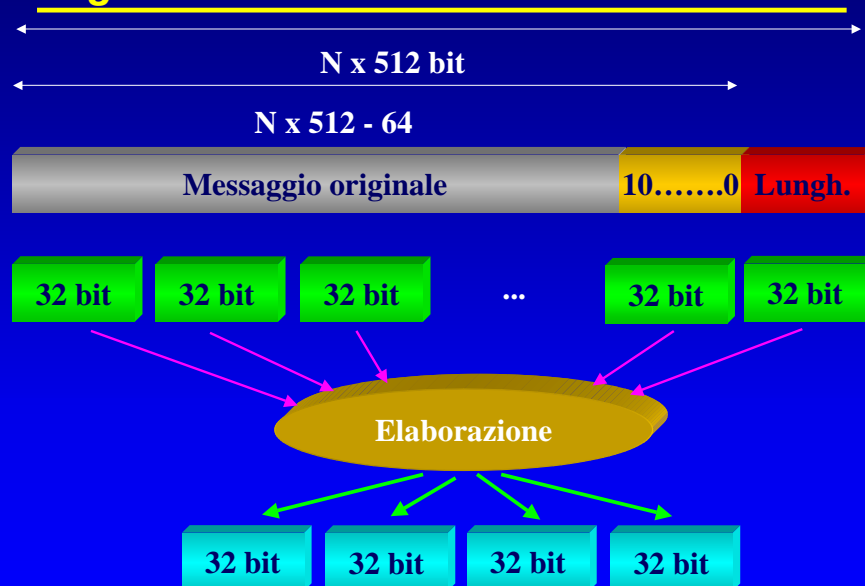
Digest

- Gli standard più usati per il *digest* attualmente sono sostanzialmente due:
 - **Secure Hash Algorithm (SHA)**: sviluppato dal NIST e rivisto successivamente e standardizzato come FIPS PUB 180-1 noto come **SHA-1**, e usa *digest* da 160 bit.
 - **MD5** definito da Ron Rivest [RFC 1321] che usa un *digest* di 128 bit.

6.41

Integrità e Firma elettronica

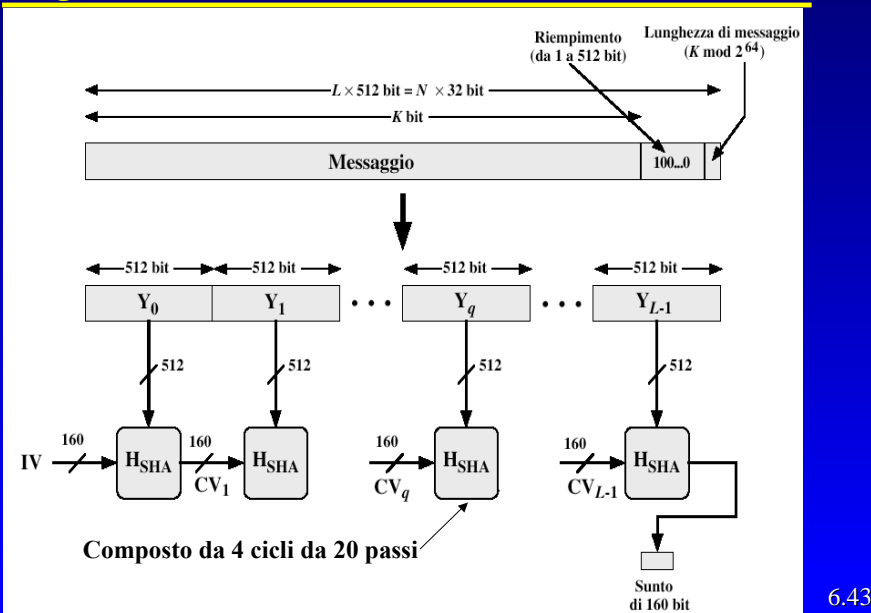
Digest - MD5



6.42

Integrità e Firma elettronica

Digest - SHA-1



6.43

Integrità e Firma elettronica

Digest – The Birthday Attack

- Quanti studenti sono necessari in una classe affinché la probabilità che almeno due di essi siano nati lo stesso giorno sia superiore a $1/2$?
 - Dalla teoria della probabilità la risposta è 23.
- In generale, se esiste una qualche relazione tra n ingressi (studenti, messaggi, ecc.) e k uscite (compleanni, digest):
 - ci sono $n(n-1)/2$ coppie di ingressi possibili;
 - se $n(n-1)/2 > k$, la probabilità di avere due ingressi che diano luogo alla stessa uscita è abbastanza elevata;
 - in pratica è probabile per $n > \sqrt{k}$
- Se m è la lunghezza del digest, tra $2^{m/2}$ diversi messaggi è abbastanza probabile trovarne due che diano luogo allo stesso digest
 - più il digest è lungo più è difficile trovare tale coppia di messaggi.

6.44

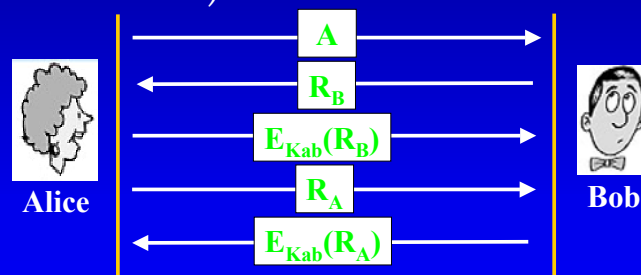
Autenticazione

- Obiettivo: Bob vuole che Alice provi la sua identità
- Una prima serie di Authentication Protocol (AP) semplici potrebbero essere:
 - **AP1:** Alice invia un pacchetto dicendo "Sono Alice".
 - **AP2:** Alice invia un pacchetto dicendo "Sono Alice" ed allega il suo numero IP.
 - **AP3:** Alice invia un pacchetto dicendo "Sono Alice" ed allega una password.
 - **AP3.1:** Alice invia un pacchetto dicendo "Sono Alice" ed allega una password cifrata.

6.45

Autenticazione

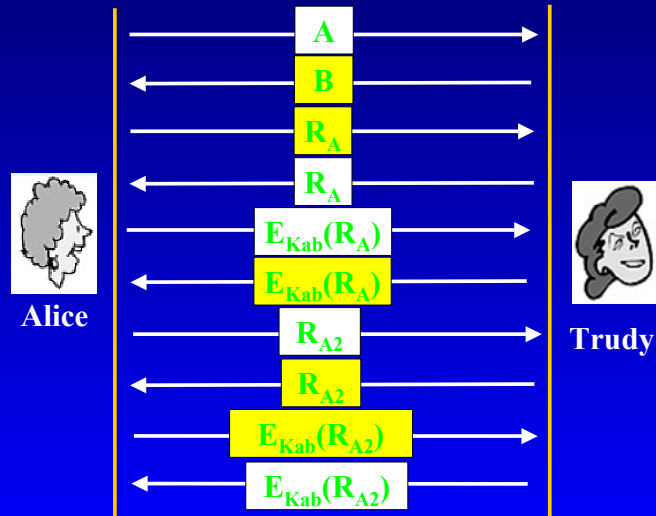
- **AP4** Per evitare un "playback attack", si usa un *nonce* (*only once*) R , ossia un numero usato una volta sola generato a caso da Bob).



- L'algorithmo è soggetto ad un "reflection attack".

6.46

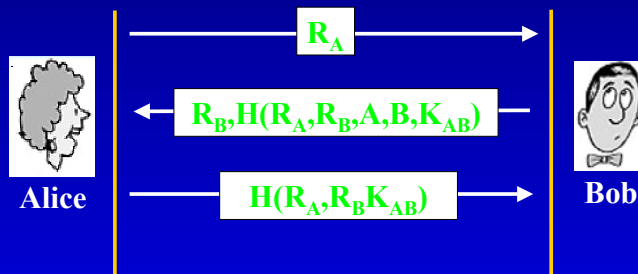
Autenticazione *Reflection Attack*



6.47

Autenticazione

- AP5 Uso di una funzione di hash $H(\cdot)$.



- Questo algoritmo è sicuro perché un intruso non può ottenere da nessuna delle due parti il digest di un messaggio desiderato
 - Entrambi i digest contengono valori scelti dal mittente, che l'intruso non può controllare.

6.48

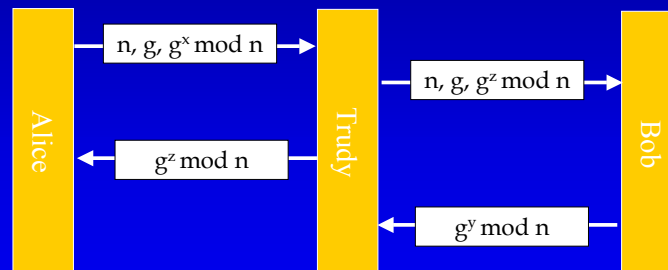
Autenticazione

- Come si scambiano la chiave pubblica?
- Una elegante soluzione consiste nell'algoritmo di Diffie-Hellman:
 - Alice genera due numeri primi molto grandi, n e g (in modo che rispettino determinate proprietà) e un qualunque numero x .
 - Alice trasmette a Bob $(n, g, g^x \bmod n)$.
 - Bob genera un numero casuale y .
 - Bob invia ad Alice $(g^y \bmod n)$ e calcola $[(g^x \bmod n)^y \bmod n]$.
 - Alice calcola $[(g^y \bmod n)^x \bmod n]$.
 - Per l'aritmetica dei moduli le due quantità sono uguali e pari a $(g^{xy} \bmod n)$; questo valore rappresenta la chiave di sessione.

6.49

Autenticazione

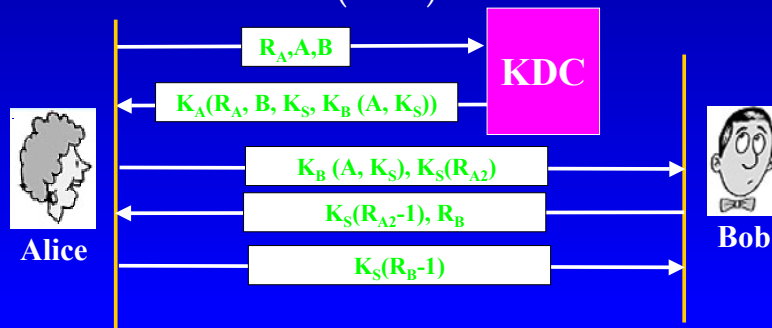
- L'algoritmo di Diffie-Helman è soggetto all'attacco *man-in-the-middle*.



6.50

Autenticazione

- Mantenere chiavi segrete differenti per tutti i sistemi con cui si ci vuole autenticare può non essere realistico.
- **AP6** Un meccanismo per ovviare a questo problema consiste nell'utilizzare un *Key Distribution Center (KDC)*.



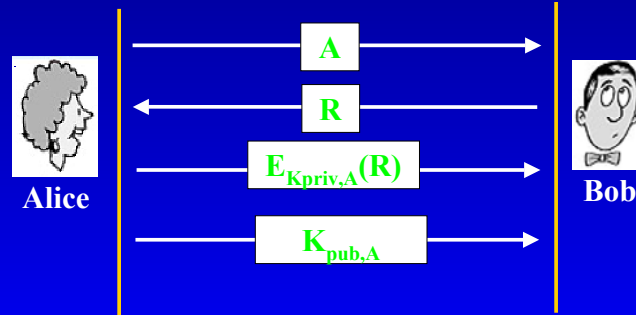
Autenticazione

- L'algoritmo di autenticazione visto è il Needham-Schroeder.
- Sebbene sembri sicuro, in realtà presenta una falla, nel caso l'intruso sia venuto in possesso di una vecchia chiave di sessione K_s . Needham e Schroeder ne hanno proposto una variante che risolve il problema.
- Questa forma di autenticazione è utilizzata dal *Kerberos* (Windows 2000):
 - prevede la presenza di un client (utente), un server di autenticazione (AS), un server emittitore di *ticket* (TGS, *Ticket Grant Server*) e i server che forniscono i servizi;
 - l'AS autentica gli utenti tramite la password e fornisce una chiave per autenticarsi presso il TGS;
 - il TGS fornisce una chiave di sessione e permette di autenticare l'utente presso i diversi server presenti;
 - le repliche dei messaggi sono evitate utilizzando un timestamp nei messaggi scambiati.

6.52

Autenticazione

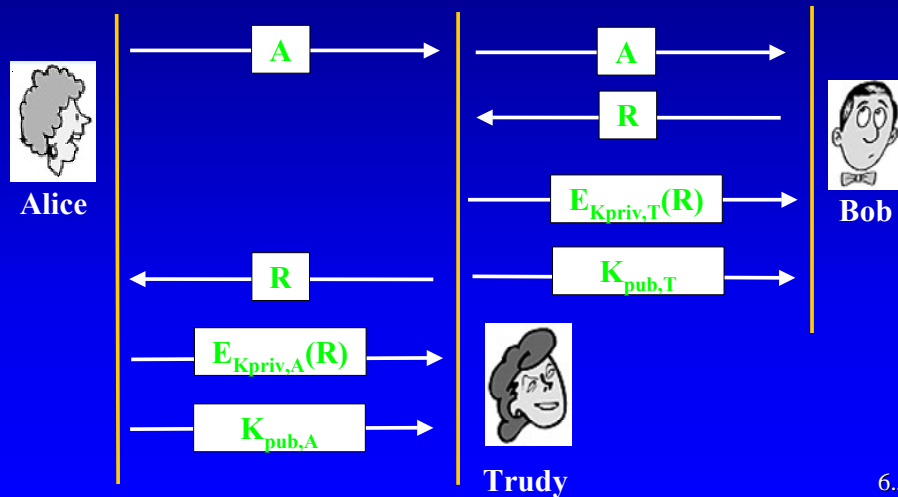
- AP7: utilizzo della chiave pubblica:



6.53

Autenticazione

- AP7 è soggetto all'attacco nel mezzo (*in the middle attack*)



6.54

Autenticazione

Distribuzione delle chiavi

- Il problema dello scambio delle chiavi pubbliche in AP7 è risolto utilizzando i certificati.
- Un certificato è un documento elettronico in cui si trovano diversi dati dell'utente (nome, indirizzo, email, sito web) e la sua chiave pubblica.
- La validità del certificato (e quindi l'appartenenza della chiave pubblica al soggetto indicato) deve essere garantita da un'autorità riconosciuta da entrambe le parti.

6.55

Autenticazione

Distribuzione delle chiavi

- I certificati, per essere validi, devono essere "firmati" da una *Certification Authority* (CA), riconosciuta come fidata da tutti
 - le entità (persone, router, etc.) possono registrare le loro chiavi pubbliche alla CA;
 - l'entità che si iscrive deve fornire una "prova dell'identità" alla CA.



- Quando Alice vuole la chiave pubblica di Bob:
 - Prende il certificato di Bob (da Bob, dalle apposite *Directory*).
 - Applica la chiave pubblica della CA e ricava la chiave pubblica di Bob.

6.56

Autenticazione Certificati X.509

- Uno degli standard più diffuso è X.509 (ITU).
- In sostanza, X.509 è una descrizione dei contenuti del certificato.
- I certificati X.509 sono codificati utilizzando ASN.1.

Version	Versione X.509
Serial Number	Identifica univocamente il certificato insieme al nome della CA
Signature Algorithm	L'algoritmo utilizzato per la firma
Issuer	Nome della CA
Validity period	Periodo di tempo in cui è valido
Subject name	Entità a cui appartiene il certificato
Public key	Chiave pubblica e algoritmo che la utilizza
Issuer ID	Un identificativo opzionale del soggetto che ha emesso il certificato
Subject ID	Un identificativo opzionale del soggetto del certificato
Extensions	Molte estensioni definite
Signature	Firma del certificato

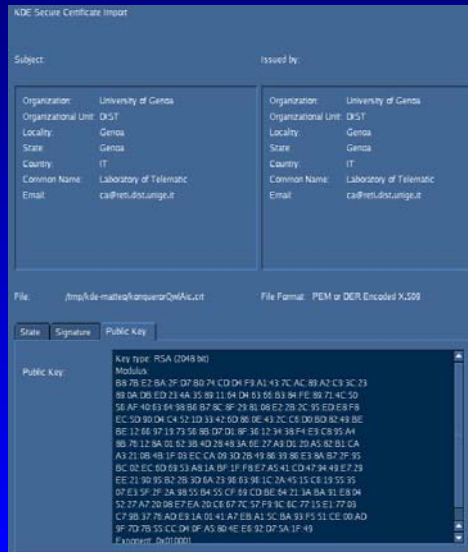
6.57

Autenticazione Certification Authority

- **Avere una sola certification authority non è possibile**
 - carico di lavoro elevato;
 - punto critico in caso di guasti;
 - difficoltà di individuazione su scala mondiale.
- **Si è pensato di utilizzare un diverso approccio, denominato PKI (*Public Key Infrastructure*):**
 - certificati;
 - molteplici CA, organizzate gerarchicamente;
 - directory, per la conservazione dei certificati in locazioni pubbliche.
- **L'organizzazione attuale prevede diverse CA all'apice della gerarchia**
 - le loro chiavi pubbliche si trovano nei browser.
- **I certificati possono essere revocati**
 - aggiornamento delle *Certificate Revocation List* da parte delle CA;
 - in caso di abusi o compromissione della segretezza.

6.58

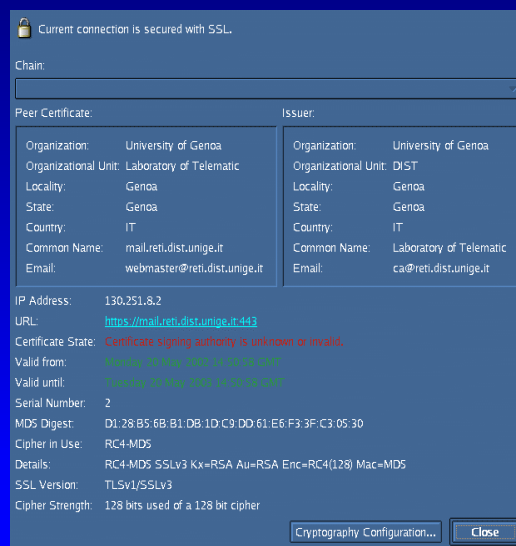
Autenticazione Certificati



Certificato di una
CA autofirmato

6.59

Autenticazione Certificati



Certificato di un
server web firmato
da una CA non
riconosciuta dal
browser

6.60

Autenticazione Certificati

Current connection is secured with SSL.

Chain:

Peer Certificate:	Issuer:
Organization: University of Genoa	Organization: University of Genoa
Organizational Unit: Laboratory of Telematic	Organizational Unit: DIST
Locality: Genoa	Locality: Genoa
State: Genoa	State: Genoa
Country: IT	Country: IT
Common Name: mail.reti.dist.unige.it	Common Name: Laboratory of Telematic
Email: webmaster@reti.dist.unige.it	Email: ca@reti.dist.unige.it

IP Address: 130.251.8.2
 URL: <https://mail.reti.dist.unige.it/>
 Certificate State: The certificate is valid.
 Valid from: Monday, 30 May 2006 14:50:58 GMT
 Valid until: Tuesday, 30 May 2006 14:50:58 GMT
 Serial Number: 2
 MD5 Digest: D1: 28 B5: 6B: B1: DB: 1D: C9: DD: 61: E6: F3: 3F: C3: 05: 30
 Cipher in Use: RC4-MD5
 Details: RC4-MD5 SSLv3 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5
 SSL Version: TLSv1/SSLv3
 Cipher Strength: 128 bits used of a 128 bit cipher

**Certificato di un
server web firmato
da una CA
riconosciuta dal
browser**

6.61

Autenticazione e segretezza

- Si osservi che la pratica usuale è quella di:
 - Usare chiave simmetriche per la cifratura dei dati (più veloci).
 - Cambiare spesso (ogni sessione o più) la chiave simmetrica.
 - Scambiarsi la chiave simmetrica tramite una cifratura a chiave pubblica.
 - Autenticare l'identità della chiave pubblica usando una CA.

6.62

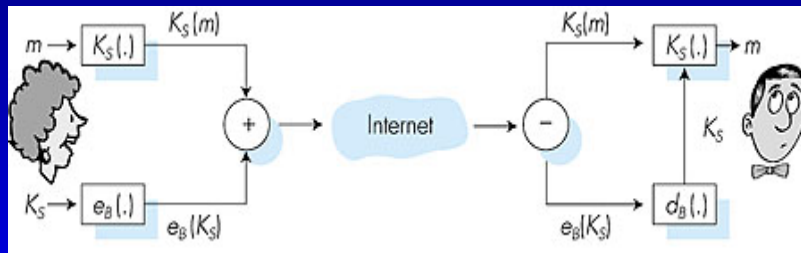
Sicurezza - Protocolli

- Oltre che dal punto di vista della locazione fisica dei meccanismi di sicurezza, riveste una notevole importanza la scelta del loro posizionamento nella pila protocollare.
- I dispositivi di sicurezza possono essere implementati:
 - A livello di applicazione (ad es. email, DNS)
 - A livello di trasporto (ad es. SSL, SET)
 - A livello di rete (IPsec)
 - A livello di linea (WEP)

6.63

E-mail sicure - Segretezza dei dati

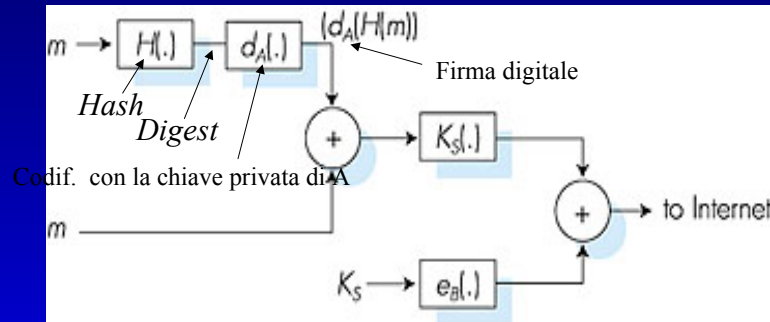
- Alice vuole inviare un messaggio m segreto a Bob



- Genera una chiave simmetrica casuale, K_S
- Cifra il messaggio con K_S , $K_S(m)$.
- Cifra anche K_S con la chiave pubblica di Bob, $e_B(K_S)$.
- Invia sia $K_S(m)$ che $e_B(K_S)$ a Bob

6.64

E-mail sicura - Segretezza, autenticazione ed integrità



- Il digest del messaggio viene cifrato con la chiave privata del mittente (firma e integrità)
- Il messaggio viene cifrato con una chiave simmetrica insieme alla firma; il tutto viene cifrato con la chiave pubblica del destinatario (segretezza)

6.65

E-mail sicura: standard

- PGP, *Pretty Good Privacy*
 - prevede autenticazione, segretezza, firma digitale e compressione;
 - gli utenti si scambiano le chiavi;
 - ad oggi è lo standard più diffuso.
- PEM, *Privacy Enhanced Mail*
 - standard IETF, RFC 1421-1424;
 - non è mai stato utilizzato (richiede una unica root per certificare).
- S/MIME, *Secure MIME*
 - si integra perfettamente con le codifiche MIME;
 - prevede una gerarchia di CA per la distribuzione di certificati con le chiavi pubbliche degli utenti.

6.66

E-mail sicura - PGP

Pretty Good Privacy (PGP)

- E' uno schema di di cifratura per e-mail, uno standard de facto.
- Usa la cifratura simmetrica (Triple-DES o IDEA) e a chiave pubblica (RSA), le funzioni di *Hash* (MD5 o SHA) e la firma digitale come descritto prima
- Quindi fornisce riservatezza, autenticazione del mittente e verifica dell'integrità del messaggio
- Inventato da Phil Zimmermann, oggetto per tre anni di indagini da parte federale (USA).

```

---BEGIN PGP SIGNED MESSAGE-
--
Hash: SHA1

Bob:My husband is out of
town tonight.Passionately
yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRhhGJGhg/12EpJ+1o8gE4vB
3mqJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---

```

6.67

Sicurezza nei DNS - DNSSec

- DNS spoofing: un intruso può inserire risoluzioni errate nelle cache dei *Name Server*.
- DNSSec è un progetto per rendere sicura la risoluzione dei nomi in Internet (RFC 2535).
- I suoi obiettivi principali sono:
 - autenticare le transazioni;
 - distribuire le chiavi pubbliche appartenenti ai domini registrati;
 - distribuzione dei certificati degli utenti.

6.68

Secure Socket Layer (SSL)

- SSL opera a livello di trasporto e fornisce funzioni per la sicurezza ad ogni applicazione basata su TCP
- E' utilizzato da varie applicazioni fra cui *www server* e *browser* per servizi di *e-commerce* (shttp)
- I servizi per la sicurezza di SSL sono:
 - Autenticazione del server (tramite certificato firmato da CA fidate)
 - Cifratura dei dati
 - Autenticazione dei client (opzionale)
- E' la base della **Transport Layer Security (TSL)** dell'IETF

6.69

Secure Socket Layer (SSL)

- La comunicazione si compone di due protocolli:
 - instaurazione della connessione, permette di autenticare il server e di stabilire una chiave di sessione;
 - trasporto dell'informazione cifrata.
- SSL supporta diversi algoritmi di cifratura:
 - DES con 3 chiavi + SHA-1 a 168 bit è la combinazione più sicura, anche se molto lenta;
 - RC4 a 128 bit + MD5 è la combinazione più usata per il commercio elettronico, è più veloce ma anche più facile da violare.

6.70

Secure Socket Layer (SSL)

Autenticazione del server

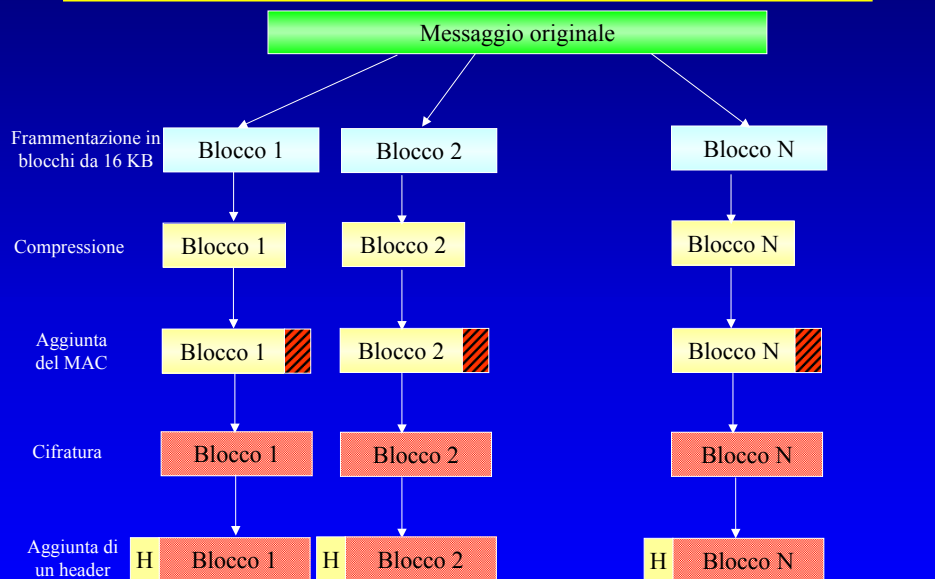
- Un *browser* con SSL deve possedere la chiave pubblica di una o più CA.
- Il *browser* richiede il certificato del Server secondo uno dei CA che conosce.
- Il *browser* usa la chiave pubblica del CA per estrarre la chiave pubblica del Server.

Sessioni SSL

- Per effettuare lo scambio sicuro, SSL crea delle sessioni che possono essere usate anche da più connessioni TCP contemporaneamente
- La sessione prevede:
 - la generazione di una chiave simmetrica da parte del *browser*, cifrata con la chiave pubblica del server e ad esso inviata;
 - La decifrazione della chiave simmetrica da parte del server
 - Uno scambio per definire se e come i messaggi verranno cifrati

6.71

Secure Socket Layer (SSL)



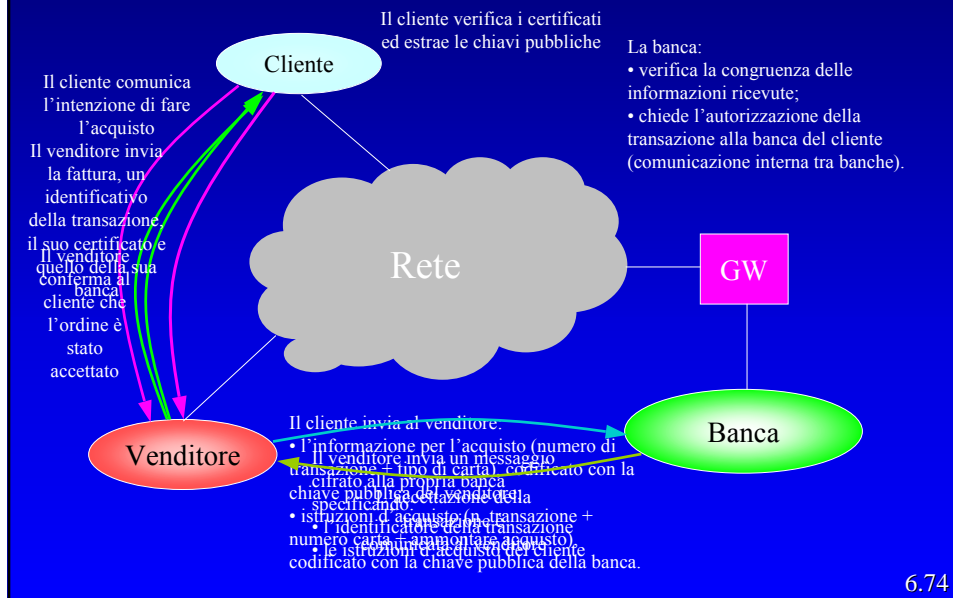
6.72

Secure Electronic Transaction (SET)

- E' stato progettato per realizzare i pagamenti via carta di credito e le transazioni via Internet (VISA e Mastercard);
- Prevede la presenza di tre attori che devono essere tutti certificati:
 - Cliente (certificato dalla propria banca)
 - Venditore (certificato dalla propria banca)
 - Banca (del venditore)
- Dà significato legale ai certificati;
- Il numero di carta di credito del cliente passa alla banca del venditore senza che quest'ultimo lo possa vedere.
- Tre componenti software:
 - *Browser wallet* (portafoglio)
 - *Merchant server*
 - *Acquirer Gateway*

6.73

Secure Electronic Transaction (SET)



6.74

Sicurezza a livello di rete IPsec (IP security)

- La cifratura continua ad essere *end-to-end* ma viene effettuata nel livello di rete sui pacchetti IP e quindi diventa disponibile a tutti i protocolli che usano IP (oltre TCP, UDP, ICMP, SNMP,...).
- Per quanto concerne l'autenticazione, in questo caso questa può avvenire anche nei confronti di indirizzi IP.
- IPsec si compone di due protocolli:
 - *Authentication Header (AH) protocol*
 - *Encapsulation Security Payload (ESP) protocol*

6.75

Sicurezza a livello di rete IPsec (IP security)

- Alcuni esempi di utilizzo di IPsec sono:
 - Interconnessione sicura di reti aziendali tramite Internet (in sostanza permette la realizzazione di *Virtual Private Network (VPN)*).
 - Accesso remoto sicuro in Internet.
 - Interconnessione sicura fra organizzazioni diverse via Internet.
 - Migliore sicurezza nel commercio elettronico.

6.76

Sicurezza a livello di rete IPsec (IP security)

- Ambedue i protocolli di IPsec (ESP e AH) operano tramite una canale logico a livello di rete chiamato **Security Association (SA)**, creato tra sorgente e destinazione con un *handshake*.
- L'SA è
 - Unidirezionale
 - Univocamente determinato da:
 - » Protocollo di sicurezza usato (ESP o AH).
 - » Indirizzo IP della sorgente.
 - » ID a 32 bit della connessione (SPI, *Security Parameter Index*).

6.77

Sicurezza a livello di rete IPsec - AH

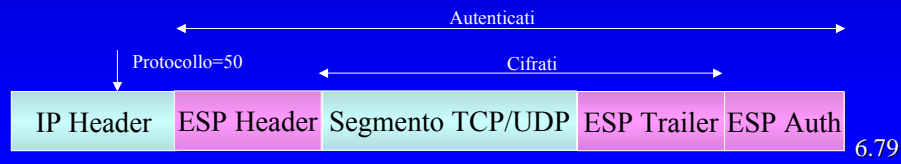
- Fornisce l'autenticazione dell'*host* e l'integrità dei dati ma non la riservatezza.
- L'intestazione AH viene inserita fra quella IP ed i dati
- Il numero di protocollo è il 51
- I *router* intermedi elaborano il *datagram* in modo usuale.
- L'intestazione dell'AH comprende:
 - un identificatore di connessione;
 - un campo che specifica il tipo di dati trasportati (UDP, TCP, ICMP...);
 - un numero di sequenza;
 - Un *digest* "firmato" e calcolato sul *datagram* originale.
- La firma di solito viene effettuata con la chiave simmetrica negoziata.



6.78

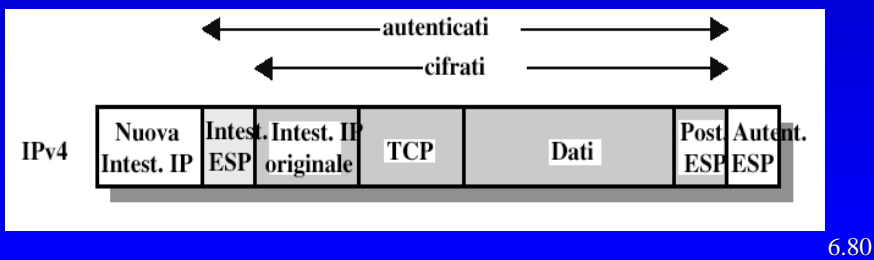
Sicurezza a livello di rete IPsec - ESP

- Fornisce la riservatezza, l'autenticazione dell'host e l'integrità dei dati
- I dati e il postambolo dell'ESP sono cifrati
- L'indicazione della successiva intestazione è nel postambolo ESP.
- L'header ESP contiene un numero di sequenza e il SPI.
- Il campo di autenticazione del ESP è simile ha quello dell'AH.
- Il numero di protocollo contenuto nell'intestazione IP quando si usa ESP è 50.



Sicurezza a livello di rete IPsec - Modalità di trasporto

- Due sono le modalità di funzionamento:
 - Modalità di trasporto
 - Modalità Tunnel
 - » applicabile se le due entità sono apparati intermedi come *router* o *firewall*.
 - » permette comunicazioni sicure a terminali che non usano IPsec.
 - » Permette la cifratura dell'intero pacchetto IP.



Sicurezza a livello di rete

IPsec - SA

- L'architettura IPsec prevede due modalità per lo scambio delle chiavi:
 - manuale, configurata dagli amministratori sugli apparati;
 - automatica, nel qual caso è necessario un meccanismo automatico per lo scambio e la gestione delle chiavi
 - » *Internet Key Exchange* (IKE, RFC 2409) è il protocollo di default per lo scambio delle chiavi dell'IPsec;
 - » *Internet Security Association and Key Management Protocol* (ISAKMP, RFC 2047 e 2048) definisce le procedure per stabilire ed interrompere gli SA. L'associazione per la sicurezza ISAKMP definisce permette di utilizzare diversi algoritmi per lo scambio delle chiavi, tra i quali IKE.

6.81

Sicurezza a livello di linea

WEP

- Le reti wireless sono per loro stessa natura particolarmente sensibili alle problematiche di sicurezza.
- L'802.11 nella sua versione originale prevede l'utilizzo della cosiddetta *Wired Equivalent Privacy* (WEP), un meccanismo che dovrebbe garantire lo stesso livello di sicurezza di una rete cablata.
- Ogni stazione condivide una chiave segreta con l'AP
 - il meccanismo di distribuzione di queste chiavi non è definito nello standard.

6.82

Sicurezza a livello di linea

WEP

- L'algoritmo utilizzato è il RC4 (Rivest, 1994).
 - al messaggio viene aggiunto un CRC;
 - il risultato viene posto in XOR con un *keystream* generato dall'algoritmo RC4 a partire dalla chiave segreta e da un vettore iniziale IV;
 - il pacchetto risultante (IV + messaggio cifrato) viene inviato sul canale.
- La chiave in genere è la stessa per tutti gli utenti
 - tutti possono vedere il traffico della rete, come in una rete Ethernet.

6.83

Sicurezza a livello di linea

WEP

- L'algoritmo non è molto sicuro:
 - è possibile riuscire a violare il codice osservando un numero di pacchetti non molto alto;
 - è perfino possibile ricavare una coppia IV-*keystream* per interferire con la comunicazione.
- L'IEEE non prevede di migliorare questo meccanismo
 - ritiene il livello di protezione sia congruente con quanto definito inizialmente;
 - per il supporto di un maggior livello di sicurezza rimanda alla definizione dello standard 802.11i.

6.84

Sicurezza a livello di linea Bluetooth

- Il range trasmissivo di Bluetooth è più limitato di 802.11
 - minori problemi di sicurezza.
- Prevede 3 livelli di sicurezza, da nessuna protezione ad una completa cifratura e controllo di integrità.
- Due device Bluetooth per comunicare devono condividere una chiave (*passkey*)
 - i dispositivi scelgono il livello di sicurezza;
 - selezionano una chiave di sessione a 128 bit;
 - la cifratura avviene con un algoritmo denominato E_0 e il controllo di integrità con l'algoritmo SAFER+.
- Il livello di protezione non è molto elevato.
- Si possono autenticare solo i terminali e non gli utenti.

6.85