

CRIPTTOGRAFIA E SICUREZZA DELLE RETI

Università degli Studi di Cassino
Laurea Specialistica in
Ingegneria delle Telecomunicazioni

Collocazione del corso

- 2° anno della Laurea Specialistica in Ingegneria delle Telecomunicazioni
- 5 CFU
- Obbligatorio per l'orientamento “Telematica”

Contenuti

- Principi introduttivi sulla crittografia e sulla sicurezza
- Tecniche di crittografia simmetrica (a chiave segreta)
- Tecniche di crittografia asimmetrica (a chiave pubblica)
- Applicazioni della crittografia alle reti:
autenticazione, segretezza, integrità, etc.
- Una parte del corso dedicata ad esercitazioni in aula informatica

Riferimenti

- W. Stallings. *Crittografia e Sicurezza delle Reti*. McGraw-Hill, 2004.
- W. Stallings. *Cryptography and Network Security, Principles and Practice*. Pearson Education, 2003.
- A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

Background

- La protezione delle informazioni ha subito negli ultimi decenni un radicale cambiamento
- Classicamente, tale protezione era garantita mediante strumenti fisici (serrature, casseforti, etc.)
- L'avvento dei computer ha richiesto l'introduzione di strategie alternative per la protezione delle informazioni.
- In più, l'uso delle reti richiede strumenti che permettano di proteggere le informazioni durante la loro trasmissione

Definizioni

- **Computer Security** – Nome generico che indica l'insieme dei tool utilizzati per proteggere i dati immagazzinati nei computer
- **Network Security** – Insieme di misure utilizzate per proteggere i dati durante la loro trasmissione, ad esempio su una LAN o su una rete telefonica
- **Internet Security** - Insieme di misure utilizzate per proteggere i dati durante la loro trasmissione sulla rete Internet

Definizioni

- **Crittografia:** si occupa di garantire la segretezza delle informazioni
- **Sicurezza:** utilizza tecniche crittografiche per garantire la “sicurezza” delle comunicazioni in presenza di potenziali avversari

Focus del corso

- In questo corso si studieranno
 - Tecniche di crittografia
 - L'applicazione di tali tecniche per conseguire la *internet security*, ovvero per scoraggiare, prevenire, rivelare e porre rimedio a eventuali violazioni della sicurezza durante la trasmissione delle informazioni

Esempi di violazioni della sicurezza

- Una comunicazione viene ascoltata in modo fraudolento (attacco passivo)
- A comunica con B spacciandosi per C
- A nega di aver inviato un precedente messaggio
- Nella comunicazione tra A e B, C intercetta i messaggi e li sostituisce con altri da esso creati (attacco attivo)
- Celare delle informazioni all'interno di una comunicazione (steganografia)

Architettura di Sicurezza OSI

- Un documento importante è la raccomandazione X.800 dell'ITU-T
Security Architecture for OSI
- Fornisce una panoramica astratta di molti concetti trattati nel corso
- Enuncia quali sono i requisiti di sicurezza che devono essere garantiti in una rete di telecomunicazioni

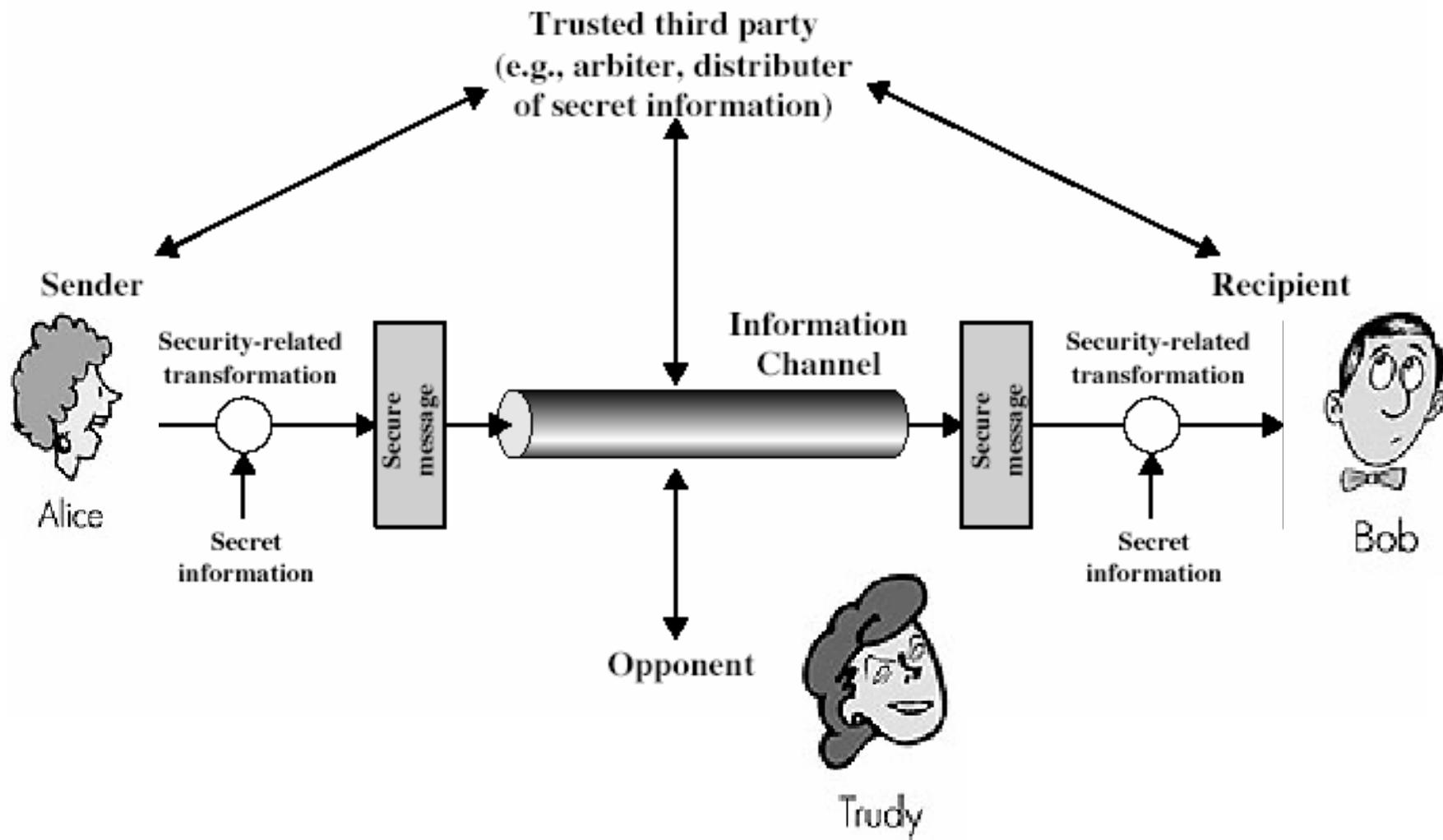
Servizi di Sicurezza in X.800

- **Autenticazione** – assicurazione dell'autenticità dei soggetti coinvolti nella comunicazione
- **Controllo degli Accessi** – Inibizione dell'uso di una risorsa da parte di soggetti non autorizzati
- **Confidenzialità** – Protezione della riservatezza dei dati
- **Integrità** – Assicurazione che i dati non siano stati alterati
- **Non-Ripudiabilità** – Protezione contro la negazione di un soggetto coinvolto nella comunicazione

Classificazione degli attacchi

- **Attacchi passivi** – ascolto celato delle comunicazioni (eavesdropping) e monitoraggio del traffico
- **Attacchi attivi** – modifica dei dati per
 - Spacciare la propria identità per un'altra
 - Rispondere a messaggi inviati ad altre parti
 - Modificare i messaggi in transito
 - Attacchi di tipo “denial of service”

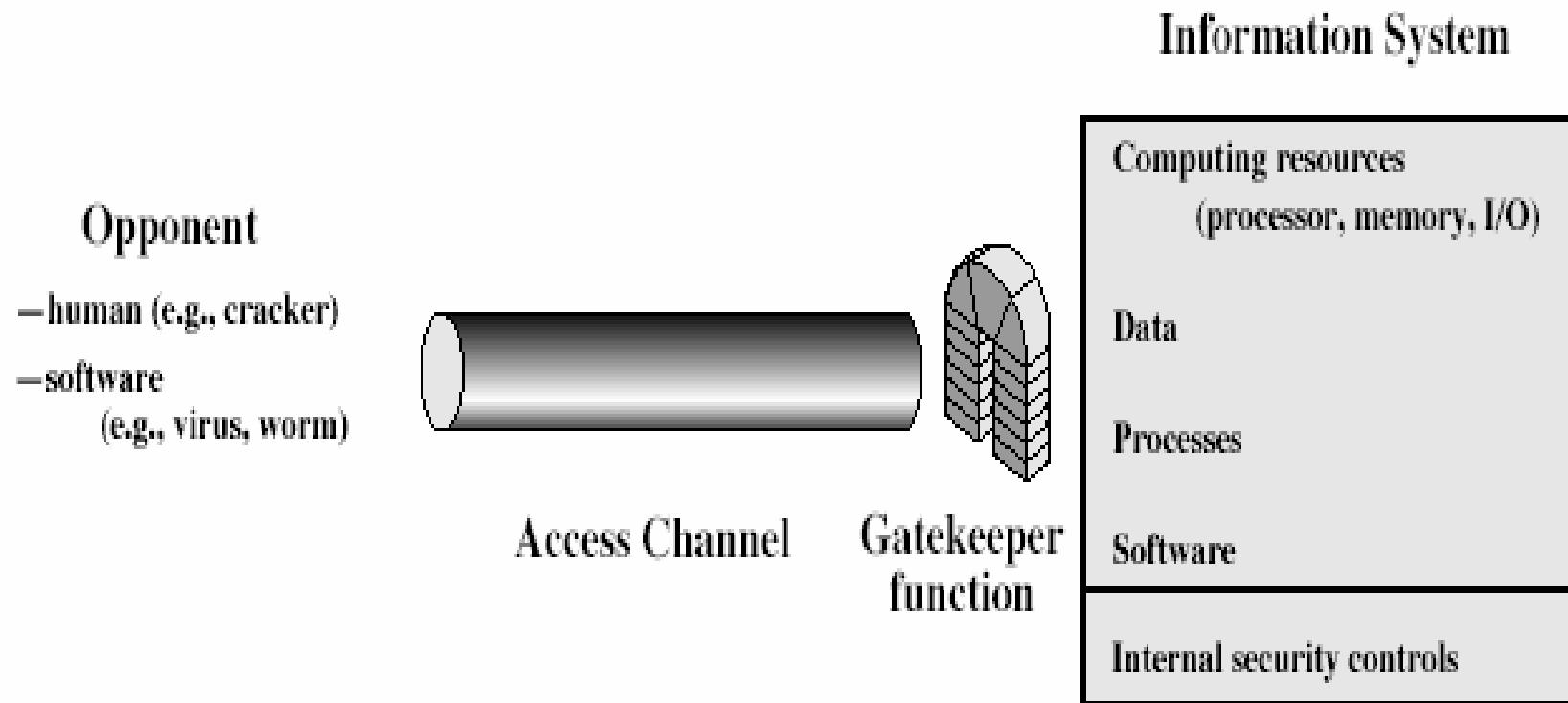
Un modello per la sicurezza delle reti



Un modello per la sicurezza delle reti

- Si evince quindi che nella progettazione di un servizio di sicurezza si deve:
 - Progettare un algoritmo per la trasformazione di sicurezza (crittografia)
 - Generare le informazioni segrete (chiavi) da utilizzare
 - Sviluppare metodi per la condivisione sicura delle chiavi
 - Specificare un protocollo (insieme di regole) che permetta ad Alice e Bob di utilizzare l'algoritmo di crittografia e le chiavi segrete per comunicare in modo sicuro

Modello di sicurezza per gli accessi alla rete



Ci occuperemo in maniera limitata di tale modello

Codifica e Crittografia: Parte I

Introduzione alla crittografia simmetrica;
Descriveremo gli algoritmi “classici” e quelli
più moderni;

In particolare, studieremo

- DES (Data Encryption System)
- AES (Advanced Encryption Standard)

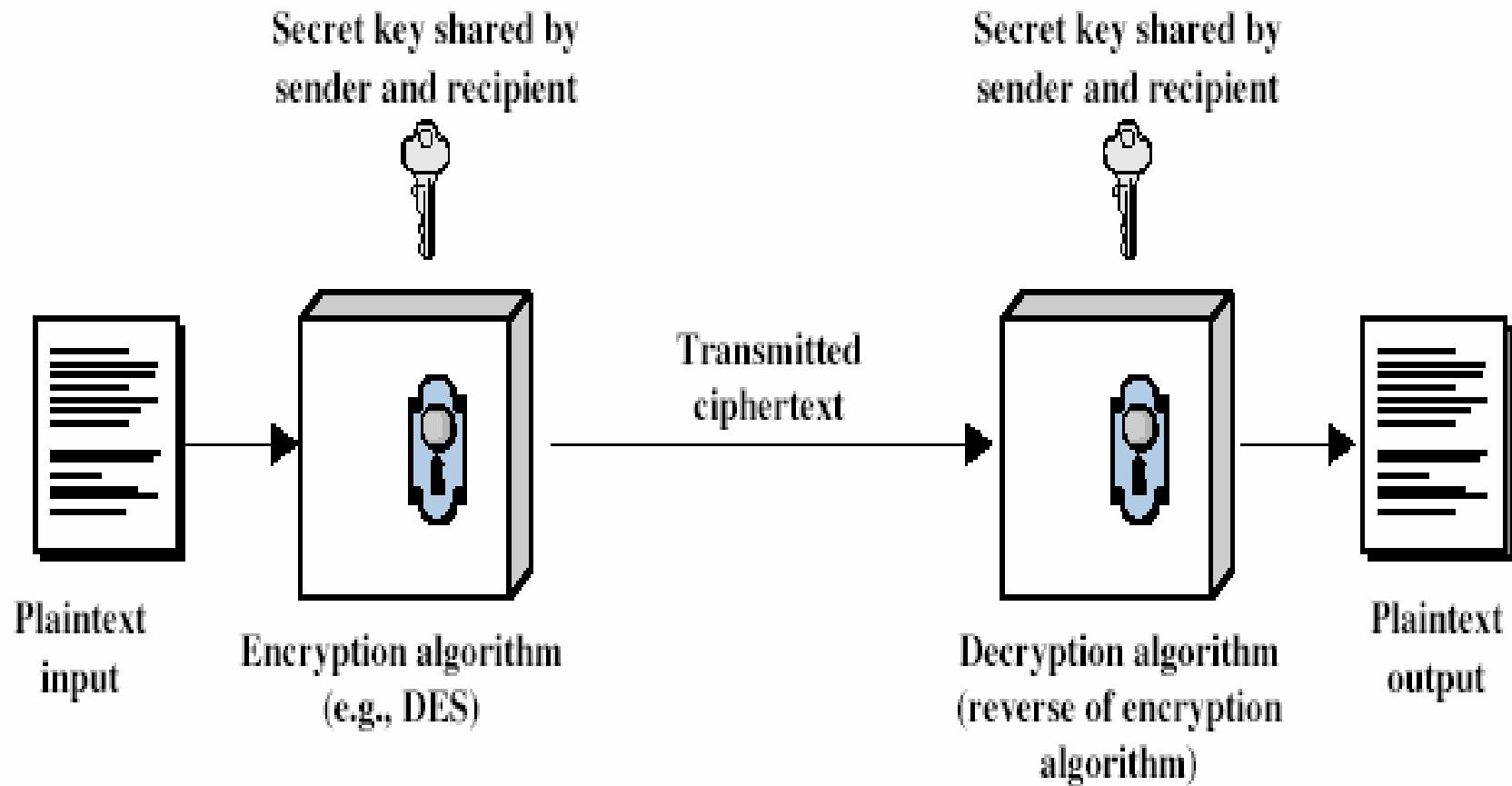
Crittografia Simmetrica

- Detta anche “a chiave privata,” “a chiave singola” o anche “convenzionale”
- Trasmettitore e Ricevitore condividono una chiave segreta
- Tutti gli algoritmi di crittografia classici sono di questo tipo
- È stata l'unico tipo di crittografia fino agli anni '70, quando fu introdotta la crittografia a chiave pubblica

Terminologia di base

- **Plaintext, testo in chiaro** – il messaggio originale
- **Ciphertext, testo cifrato** – il messaggio crittografato
- **Cipher, cifrario** – algoritmo che trasforma il plaintext in ciphertext
- **Key, chiave** – informazione usata nel cifrario, nota solo a Bob e Alice
- **encipher (encrypt), criptare** – convertire plaintext in ciphertext
- **decipher (decrypt), decriptare** – recuperare il plaintext dal ciphertext
- **crittografia** – studio dei metodi e dei principi alla base degli algoritmi di cifratura
- **cryptanalysis (codebreaking)** – studio dei principi di base e dei metodi per ottenere il testo in chiaro senza conoscere la chiave

Schema della Crittografia Simmetrica



Definizioni

$$\begin{array}{lll} X = (X_1, \dots, X_l) & \text{plaintext} & (X_i \in \mathbb{A}) \\ Y = (Y_1, \dots, Y_m) & \text{ciphertext} & (Y_i \in \mathbb{A}) \\ K = (K_1, \dots, K_n) & \text{chiave} & (K_i \in \mathbb{A}) \end{array}$$

Dati l'insieme \mathcal{K} delle chiavi e \mathcal{P} dei plaintext, si possono definire delle funzioni di **encryption**

$$E : \mathcal{K} \times \mathcal{P} \rightarrow \mathcal{C}$$

e di **decryption**

$$D : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{P}.$$

Useremo le notazioni

$$E_K[X] \qquad D_K[Y]$$

Requisiti

- Affinchè la crittografia simmetrica sia sicura è richiesto:
 - Un algoritmo di cifratura **forte**
 - Una chiave segreta nota solo al trasmettitore e al ricevitore
- Si assume che gli algoritmi $E(\cdot)$ e $D(\cdot)$ siano noti
- C'è bisogno quindi di una metodologia sicura per distribuire la chiave

Proprietà

1. Per ogni chiave K $D_K = E_K^{-1}$

cioè

$$D_K[E_K[X]] = X.$$

Quindi, per ogni K , E_K deve essere iniettiva, cioè

$$\forall X, X' \text{ con } X \neq X' \quad E_K[X] \neq E_K[X'].$$

2. Data la chiave K , sia E_K che D_K devono essere *facili* da calcolare.
3. Viceversa, dato solo il ciphertext $Y = E_K[X]$, deve essere *difficile* risalire a X e a K .

L'algoritmo di Crittografia

- È caratterizzato da:
 - Tipo di operazioni svolte nella cifratura
 - Sostituzione / Trasposizione
 - Numero di chiavi usate
 - Singola (privata) / doppia (pubblica)
 - Modalità di elaborazione del plaintext
 - blocco / flusso (una lettera per volta)

Criptoanalisi (Codebreaking)

- Può essere essenzialmente basata su
 - Analisi crittografica
 - Attacco a forza bruta

Tipi di Attacchi Criptoanalitici

- **ciphertext only**
- **known plaintext**
- **chosen plaintext**
- **chosen ciphertext**

L'attacco “ciphertext only” è il più complicato da attuare.

Attacco a Forza Bruta

- Utilizza ogni possibile chiave
- Costo proporzionale alla cardinalità dello spazio delle chiavi
- Si assume che il plaintext sia riconoscibile

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/ μ s	Time required at 10^6 encryptions/ μ s
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6.4 \times 10^{12}$ years	6.4×10^6 years

Esempio: attacco a DES

Il Cifrario a chiave singola DES è stato introdotto nel 1976 ed utilizza chiavi di 56 bit.

Nel 1976, si valutava sufficiente tale lunghezza:
Assumendo 1cifratura/ μ s, ci vorrebbero in media più di 1000 anni per trovare la chiave

Attualmente, hardware dedicati ad architettura parallela dal costo ragionevole, eseguono 1.000.000 di cifrature al μ s, ragion per cui il tempo medio di ricerca della chiave si riduce a circa 10 ore

DES quindi non è più considerato sicuro!

Un algoritmo Crittografico è...

- **incondizionatamente sicuro**
 - Indipendentemente dalle risorse computazionali a disposizioni, il cifrario non può essere rotto

In termini probabilistici, schematizzando il plaintext X ed il ciphertext Y come vettori aleatori, deve essere

$$\forall x, y \quad \mathbb{P}[X = x | Y = y] = \mathbb{P}[X = x]$$

Un algoritmo Crittografico è...

- **computazionalmente sicuro**
 - Lo sforzo computazionale richiesto per forzare il cifrario eccede il valore dell'informazione, o richiede un tempo che supera il tempo in cui l'informazione ha valore

Ovviamente, algoritmi computazionalmente sicuri oggi non è detto che lo siano in futuro!!

Crittografia classica a sostituzione

- Le lettere del testo in chiaro sono sostituite da altre lettere, o da numeri o da simboli
- Se, in alternativa, il testo è codificato in stringhe di bit, allora sequenze di bit di opportuna lunghezza saranno sostituite da altre sequenze di bit

Cifratura di Cesare

- E' il cifrario a sostituzione più antico a noi noto
- Ideato da Giulio Cesare
- Utilizzato per le comunicazioni militari
- Sostituisce ogni lettera con quella che viene 3 posizioni dopo
- esempio:

Alea iacta est

DOHD NDFWD HVZ

Cifratura di Cesare

- Banalmente, la funzione di criptazione è:

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
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- Matematicamente, si assegni ad ogni lettera un numero

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

- da cui:

$$C = E(p) = (p + k) \bmod 26$$

$$p = D(C) = (C - k) \bmod 26$$

Analisi del cifrario di Cesare

- Soltanto 26 diverse possibilità
 - A può essere codificata con A,B,..Z
- Ovvero, solo 26 possibili chiavi
- Vulnerabilità all'**attacco a forza bruta**
- Bisogna tuttavia avere la capacità di riconoscere il plaintext

Cifrari Monoalfabetici

- Generalizzazione del cifrario di Cesare
- Ciascuna lettera del testo in chiaro viene associata ad una lettera scelta a caso
- La chiave è lunga in tal caso 26 lettere

Plain: abcdefghijklmnopqrstuvwxyz

Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext: ifwewishtoreplaceletters

Ciphertext: WIRFRWAJUHYFTSDVFSUUUFYA

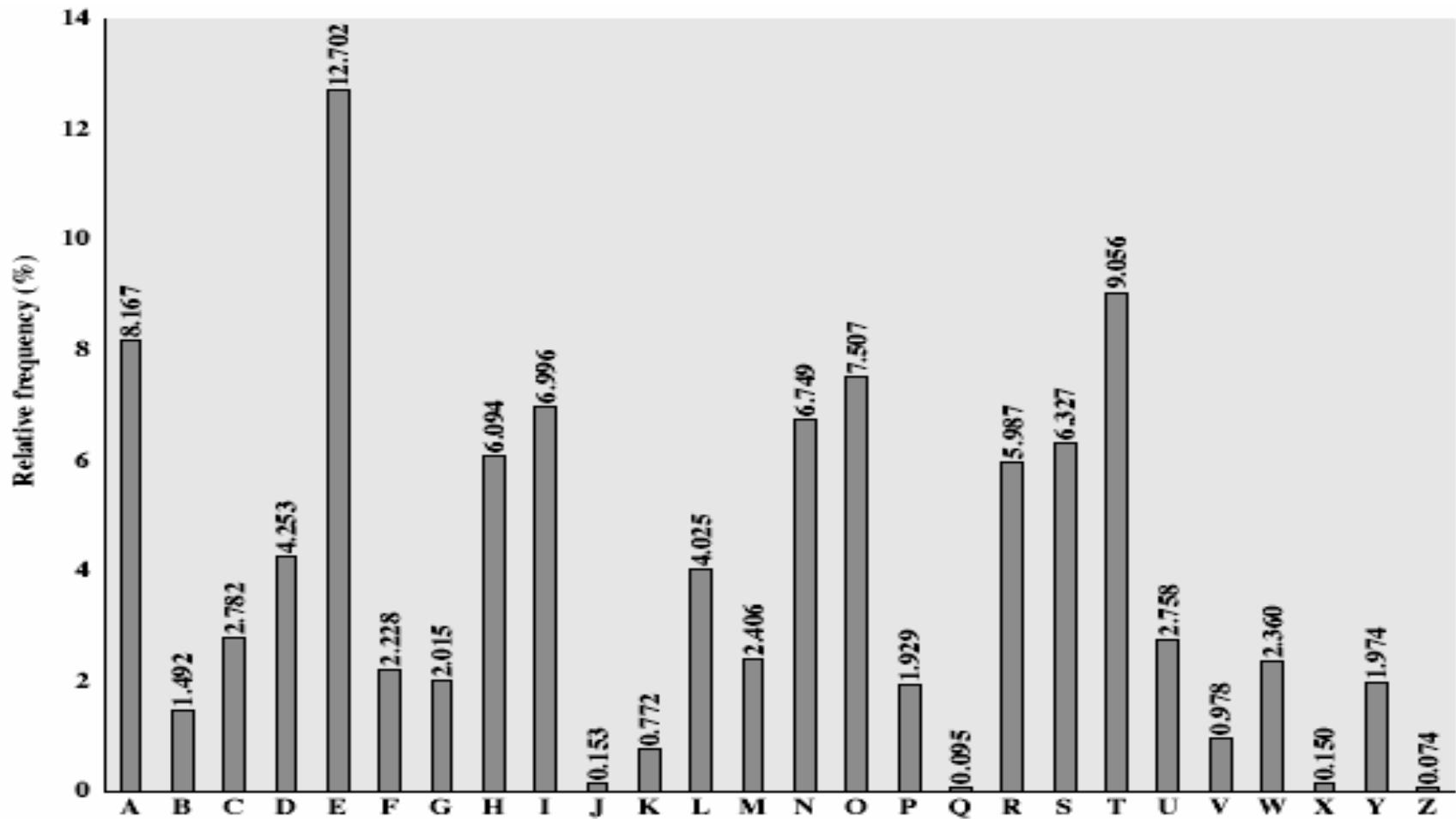
Sicurezza del cifrario monoalfabetico

- Vi sono ora $26! = 4 \times 10^{26}$ possibili chiavi
- Si potrebbe quindi pensare che tale cifrario è sicuro
- **ERRATO**
- Il cifrario si può rompere in virtù delle caratteristiche del linguaggio

Ridondanza nel Linguaggio e Criptoanalisi

- I linguaggi umani sono **ridondanti**
- Per l'inglese i digrammi “th” “sh”
- Nell'inglese **e** è la lettera più comune
- poi T,R,N,I,O,A,S
- Altre lettere sono estremamente rare:
Z,J,K,Q,X
- Vi sono tavelle di frequenza di lettere,
digrammi, trigrammi, etc...

Frequenza Alfabeto Inglese



Criptanalisi Cifrari Monoalfabetici

- Concetto base – la sostituzione monoalfabetica non modifica la frequenza relativa delle lettere
- Scoperta di matematici arabi del IX secolo
- Occorre calcolare le frequenze relative delle lettere del testo cifrato...
- ...e andare alla ricerca dei massimi e minimi
- Si individuano quindi alcune lettere
 - Un aiuto è costituito dalle tavelle dei digrammi e trigrammi più comuni

Esempio di Crittoanalisi

- Si consideri il testo cifrato:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- Si determinano le frequenze relative
- Si ipotizza che P e Z sono *e* and *t*
- Si ipotizza che ZW è *th* e quindi ZWP è *the*
- Andando avanti per tentativi si trova:

it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow

Cifratura Playfair

- Il cifrario monoalfabetico non è quindi sicuro
- Un'alternativa è cifrare i digrammi invece che le singole lettere
- Esempio: **Cifratura Playfair**
- inventata da Charles Wheatstone nel 1854, trae il suo nome dal suo amico Baron Playfair

Chiave Matriciale del Codice Playfair

- Si consideri una matrice 5X5 di lettere basate su una parola
- La parola è inserita nella matrice, la quale è poi completata con le rimanenti lettere dell'alfabeto
- I e J occupano la stessa posizione
- Esempio: si consideri la parola MONARCHY

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Criptazione e Decrittazione

- Il plaintext è criptato a digrammi:
 1. In presenza di una doppia lettera, si inserisce una lettera spuria come 'X', ovvero "balloon" diventa "ba lx lo on"
 2. Se entrambe le lettere sono sulla stessa riga, ogni lettera è sostituita da quella alla sua destra ciclica, ad esempio "ar" viene criptato in "RM"
 3. Se entrambe le lettere sono sulla stessa colonna, ogni lettera è sostituita da quella al di sotto, ad esempio "mu" viene criptato in "CM"
 4. Altrimenti, ogni lettera è sostituita da quella nella sua riga nella colonna dell'altra lettera della coppia. Ad esempio, "hs" viene criptato in "BP".

Sicurezza del Cifrario Playfair

- La sicurezza è molto maggiore del cifrario monoalfabetico
- Vi sono infatti $26 \times 26 = 676$ digrammi
- L'analisi richiede quindi una tabella con le frequenze di 676 digrammi (invece che di 26 lettere nel caso monoalfabetico)
- L'analisi richiede una quantità maggiore di testo cifrato
- Usato ampiamente, ad esempio nella I GM
- Non è tuttavia sicuro, perché conserva fortemente la struttura del plaintext

Cifrario di Hill

- Codifica m lettere di plaintext con m lettere di ciphertext

$$\begin{cases} C_1 &= (K_{11} \cdot p_1 + \cdots + K_{1m} \cdot p_m) \bmod 26 \\ \vdots \\ C_m &= (K_{m1} \cdot p_1 + \cdots + K_{mm} \cdot p_m) \bmod 26. \end{cases}$$

Cifrario di Hill

- Posto

$$C = \begin{pmatrix} C_1 \\ \vdots \\ C_m \end{pmatrix}, \quad K = \begin{bmatrix} K_{11} & \cdots & K_{1m} \\ \vdots & \ddots & \vdots \\ K_{m1} & \cdots & K_{mm} \end{bmatrix}, \quad P = \begin{pmatrix} p_1 \\ \vdots \\ p_m \end{pmatrix}$$

la cifratura/decifratura si scrive come

$$\begin{cases} E_K[P] = C = K \times P \\ D_K[C] = K^{-1} \times C = K^{-1} \times K \times P = P \end{cases}$$

ove tutte le operazioni sono intese mod26

Cifrario di Hill

- La matrice K^{-1} è l'inversa mod26 della matrice K , ovvero
$$(K \times K^{-1}) \text{ mod } 26 = (K^{-1} \times K) \text{ mod } 26 = I$$
- Se la lunghezza del blocco m è sufficientemente estesa il cifrario è sicuro ad attacchi *ciphertext only*

Attacco al Cifrario di Hill

- Vulnerabilità all'attacco *known plaintext*:

Se

$$\langle P_1, C_1 = K \times P_1 \rangle$$

⋮

$$\langle P_m, C_m = K \times P_m \rangle$$

Posto

$$P^* = [P_1 | \dots | P_m]$$

$$C^* = [C_1 | \dots | C_m]$$

è $C^* = K P^* \text{mod} 26$, da cui $K = C^* (P^*)^{-1}$

Cifrario di Hill: Esempio

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \quad K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

$$KK^{-1} = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} \text{ mod } 26 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Cifrario di Hill: Esempio di attacco

$$\begin{bmatrix} 15 & 2 \\ 16 & 5 \end{bmatrix} = K \begin{bmatrix} 5 & 8 \\ 17 & 3 \end{bmatrix} \text{ mod } 26$$

$$\begin{bmatrix} 5 & 8 \\ 17 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 9 & 2 \\ 1 & 15 \end{bmatrix}$$

$$K = \begin{bmatrix} 15 & 2 \\ 16 & 5 \end{bmatrix} \begin{bmatrix} 9 & 2 \\ 1 & 15 \end{bmatrix} = \begin{bmatrix} 137 & 60 \\ 149 & 107 \end{bmatrix} \text{ mod } 26 = \begin{bmatrix} 7 & 8 \\ 19 & 3 \end{bmatrix}$$

Cifrari Polialfabetici

- Per migliorare la sicurezza, si possono usare più cifrari monoalfabetici contemporaneamente
- Si parla in tal caso di **cifrari a sostituzione polialfabetica**
- La distribuzione dei simboli diventa maggiormente uniforme: criptoanalisi più difficile
- Viene utilizzata una chiave per selezionare il particolare cifrario monoalfabetico da utilizzare per criptare il generico simbolo
- I vari cifrari monoalfabetici sono usati ciclicamente

Cifrario di Vigenère

- E' il più semplice cifrario a sostituzione polialfabetica
- Corrisponde ad un cifrario di Cesare multiplo
- Opera su blocchi di testo lunghi m
- La chiave è costituita da una successione di lettere $K = k_1 \ k_2 \ \dots \ k_m$
- L' i -esima lettera definisce il valore dello shift sull' i -esimo carattere di plaintext
- La decriptazione è ovviamente analoga

Cifrario di Vigenère: Esempio

- Anzitutto, si scrive per esteso il plaintext
- Si scrive la chiave ripetutamente sopra il plaintext
- Si utilizza ogni lettera della chiave come una singola chiave del cifrario Cesare
- Esempio:

chiave: S I C U R E Z Z A S I C U R E Z Z A S I C U R

plaintext: V I G E N E R E N O N E M O L T O S I C U R O

ciphertext: N Q I Y E I Q D N G V G G F P S N S A K W L F

Cifrario di Vigenère: Modello Matematico

- Sia m la lunghezza della chiave, e siano P_i, C_i , e K_i la i -esima lettera dei vettori P , C e K , rispettivamente.
- Le funzioni $E_K(\cdot)$ e $D_K(\cdot)$ si esprimono come

$$\begin{aligned} E_K[P_i] &= P_i + K_i \bmod m \quad \bmod 26 \\ D_K[C_i] &= C_i - K_i \bmod m \quad \bmod 26 \end{aligned}$$

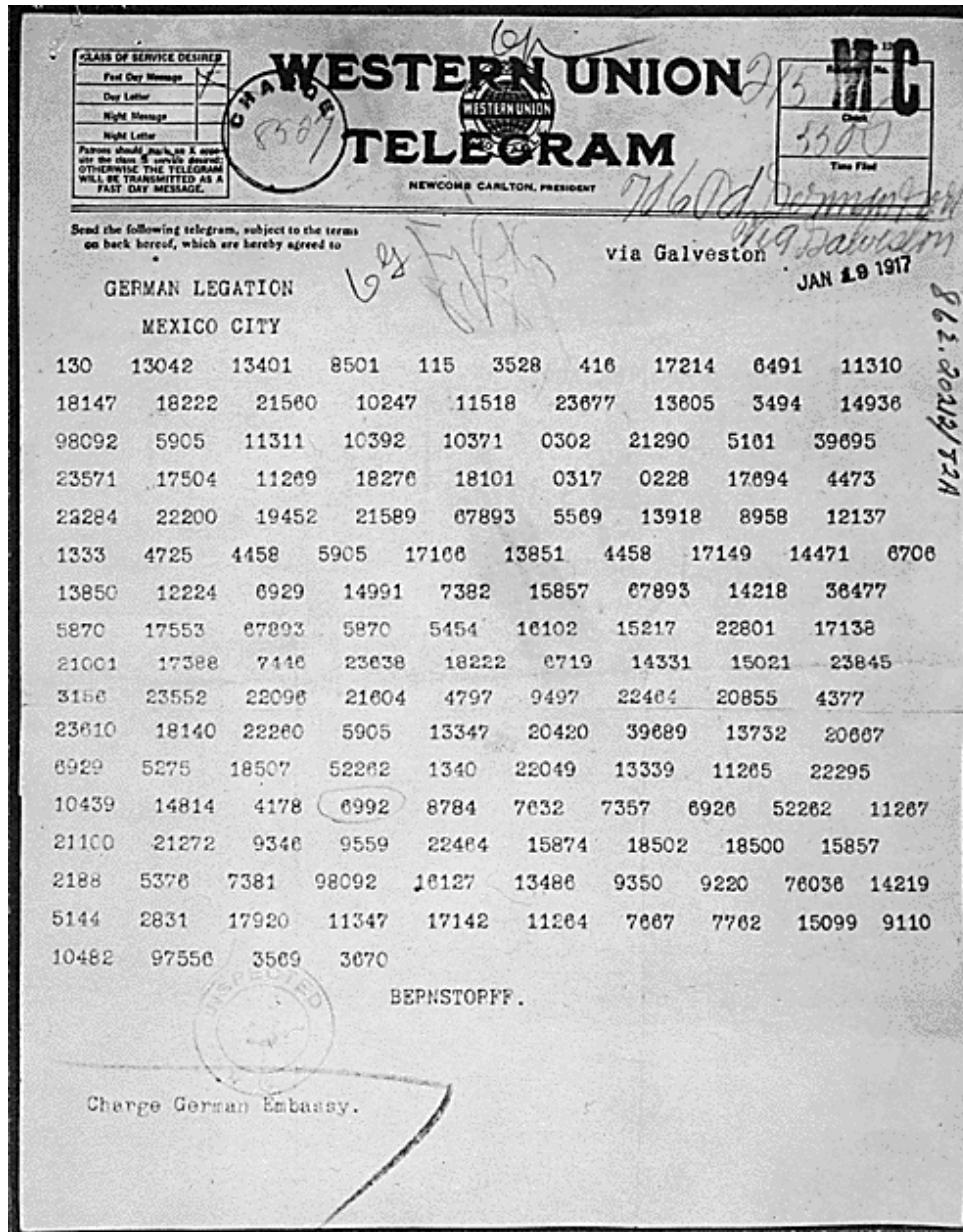
Sicurezza del cifrario Vigenère

- Ogni lettera del plaintext **non** è cifrata sempre con lo stesso simbolo
- Le frequenze relative di occorrenza delle lettere sono quindi mascherate...
- ...ma non completamente!
- Anzitutto, si verificano le frequenze nel ciphertext
 - Si verifica quindi se il cifrario è monoalfabetico
- Se non lo è, bisogna individuare m e poi operare sui singoli cifrari monoalfabetici

Sicurezza del cifrario Vigenère

- Per anni il cifrario Vigenère fu ritenuto “*le chiffre indéchiffrable*”
- Nel 1917 un articolo di *Scientific American* lo ha definito “impossibile da tradurre”
- Di fatto, era stato già violato nel 1854 da Charles Babbage e nel 1863 da Friedrich Kasiski
- Malgrado ciò, cifrari polialfabetici sono stati usati anche agli inizi del XX secolo

Il Telegramma Zimmerman



Il Telegramma Zimmerman

- Inviato nel 1917 dal Ministro degli Esteri tedesco Arthur Zimmerman all'ambasciatore tedesco in Messico su un cavo transatlantico, ritenuto sicuro perché riservato per condurre negoziati di pace
- Proponeva al Messico un'alleanza contro gli Stati Uniti promettendo in cambio denaro e delle terre del New Mexico, Arizona e Texas
- L'intento era di creare un fronte della guerra nelle Americhe al fine di distrarre forze dal fronte europeo

Il Telegramma Zimmerman

- Il telegramma fu intercettato dall'intelligence inglese e reso noto all'opinione pubblica il 24 febbraio 1917
- Il Messico immediatamente negò di aver dato seguito a quella proposta
- Il 6 Aprile 1917 il Congresso Americano approvò (con un solo voto contrario) la decisione del Presidente Wilson di entrare in guerra contro la Germania

Kasiski Method

- Svilupato da Babbage/Kasiski; risale al 1863
- Segmenti identici di plaintext producono lo stesso ciphertext se la loro distanza è multipla di m
- Digrammi e trigrammi ripetuti nel ciphertext occorrono a distanze che con elevata probabilità sono multiple di m
- Si può quindi congetturare che m sia un divisore del massimo comun divisore di tali distanze

Indice di Coincidenza

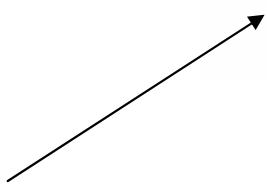
- Definito da Wolfe Friedman nel 1920
- Sia X una stringa di n caratteri e sia f_i il numero di occorrenze della lettera i in X , per $0 \leq i \leq 25$
- Si definisce indice di coincidenza in X la probabilità che due elementi di X scelti a caso siano uguali
- E' possibile estrarre due elementi da X in

$$\binom{n}{2} = n(n-1)/2 \quad \text{modi}$$

Indice di Coincidenza

- Ma allora, per ogni i , ci sono $f_i(f_i - 1)/2$ modi diversi di estrarre due caratteri entrambi uguali ad i . Quindi, si ha

$$I_c(x) = \frac{\sum_{i=0}^{25} f_i (f_i - 1)}{n(n - 1)}.$$



Prob. di estrarre due lettere uguali dal testo x

Indice di Coincidenza

- Se X è un testo inglese, le lettere A,B,... Si presenteranno con probabilità p_0, p_1, \dots, p_{25} . Se X è abbastanza lungo, si ha

$$f_i/n \approx (f_i - 1)/(n - 1) \approx p_i$$

da cui

$$I_c(x) \approx \sum_{i=0}^{25} p_i^2 = 0,065$$

Indice di Coincidenza

- In una stringa con distribuzione uniforme, ogni lettera ha probabilità 1/26, e l'indice di coincidenza sarà

$$I_c(x) \approx 26 \left(\frac{1}{26} \right)^2 = \frac{1}{26} = 0,038$$

- Tali considerazioni portano ad una metodologia per determinare m

Indice di Coincidenza

- Sia Y una stringa di testo cifrato lunga n .
- Si assuma n multiplo di m
- Si formino i vettori Y_0, Y_1, \dots, Y_{m-1} decimando con passo m il vettore Y
- Se m è effettivamente la lunghezza della chiave ciascun vettore Y_i rappresenta il ciphertext di un cifrario monoalfabetico. Di conseguenza, gli indici di coincidenza di ciascun vettore devono essere prossimi a 0,065
- In caso contrario, gli indici di coincidenza si avvicineranno a 0,038
- Mediante tentativi successivi si risale dunque al valore di m

Cifrario con autochiave

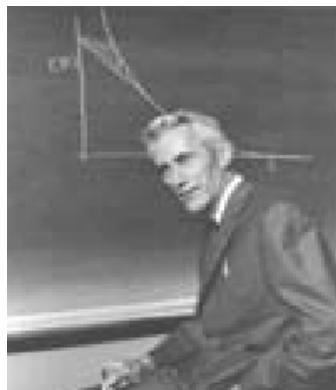
- Idealmente, si vorrebbe avere una chiave lunga quanto il messaggio
- Vigenère propose il cifrario con **autochiave**
- La chiave viene anteposta al messaggio che, ritardato opportunamente, viene usato come chiave
- Anche in questo caso si ha vulnerabilità all'analisi statistica
- Esempio con chiave *deceptive*

key: deceptivewearediscoveredsav

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

Cifrari incondizionatamente sicuri



Claude Elwood Shannon
1916-2001



- ‘*A Symbolic Analysis of Relay and Switching Circuits*’
Massachusetts Institute of Technology, Ph.D. Thesis, 1940
- ‘*A Mathematical Theory of Communication*’
Bell System Technical Journal, Vol.27, 1948, pp. 379-423; 623-656
- ‘*Communication Theory of Secrecy Systems*’
Bell System Technical Journal, Vol.28, 1949, pp. 656-715
- ‘*Memory Requirements in a Telephone Exchange*’
Bell System Technical Journal, Vol.29, 1950, pp. 343-349

Cifrari incondizionatamente sicuri

- Siano M , K e C tre variabili aleatorie che rappresentano il messaggio in chiaro, la chiave ed il testo cifrato

M prende valori in \mathcal{M} ,

K prende valori in \mathcal{K} .

C prende valori in \mathcal{C} ,

- Il cifrario è perfetto se

$$\mathbb{P}[M = m \mid C = c] = \mathbb{P}[M = m]$$

Cifrari incondizionatamente sicuri

TEOREMA: In ogni cifrario perfetto, $|K| \geq |M|$

PROVA: Poichè $E(\cdot)$ è iniettiva, deve essere $|M| \leq |C|$, ovvero il numero di testi cifrati deve essere non inferiore al numero di testi in chiaro.

Se fosse $|K| < |M| \leq |C|$, fissato un messaggio $m \in M$, esisterebbe almeno un $c^* \in C$ che non può essere generato da m , ovvero

$$P(m | c^*) = 0 \neq P(m)$$

Tale relazione contraddice il fatto che il cifrario sia perfetto

c.v.d.

One-Time Pad

- Usando una chiave “aleatoria” lunga quanto il messaggio si ottiene un cifrario perfetto
- Il cifrario è detto **One-Time pad**
- È un’evoluzione del cifrario di Vernam (1918)
- Non si può rompere perchè il testo cifrato non ha alcuna dipendenza statistica dal testo in chiaro poichè per ogni coppia (testo in chiaro, testo cifrato) esiste una chiave che realizza tale corrispondenza

One-Time Pad

- La chiave può essere usata solo una volta
- Infatti, se la chiave è lunga N bit e la uso due volte in tempi diversi, di fatto ho inviato $2N$ bit con una chiave lunga N ; in tal caso ho che $|K| < |M|$ e il cifrario perde le caratteristiche di sicurezza!
- Il problema è la distribuzione della chiave!!

Cifrari a trasposizione

- Consideriamo ora i cifrari classici a **trasposizione o permutazione**
- Il messaggio è cifrato cambiando posizione alle lettere
- La frequenza relativa delle lettere non cambia

Cifrario Rail Fence (staccionata)

- Il messaggio è scritto in diagonale per un certo numero di righe
- E poi viene letto per righe
- esempio:

m e m a t r h t g p r y
e t e f e t e o a a t

- Il ciphertext è

MEMATRHTGPRYETEFETEOAAT

Cifrario a trasposizione di righe

- È uno schema più complesso
- Il messaggio è scritto per righe in una matrice
- Poi le colonne sono lette in un ordine dettato da una certa chiave

Key: 3 4 2 1 5 6 7

Plaintext: a t t a c k p
 o s t p o n e
 d u n t i l t
 w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Cifrari composti

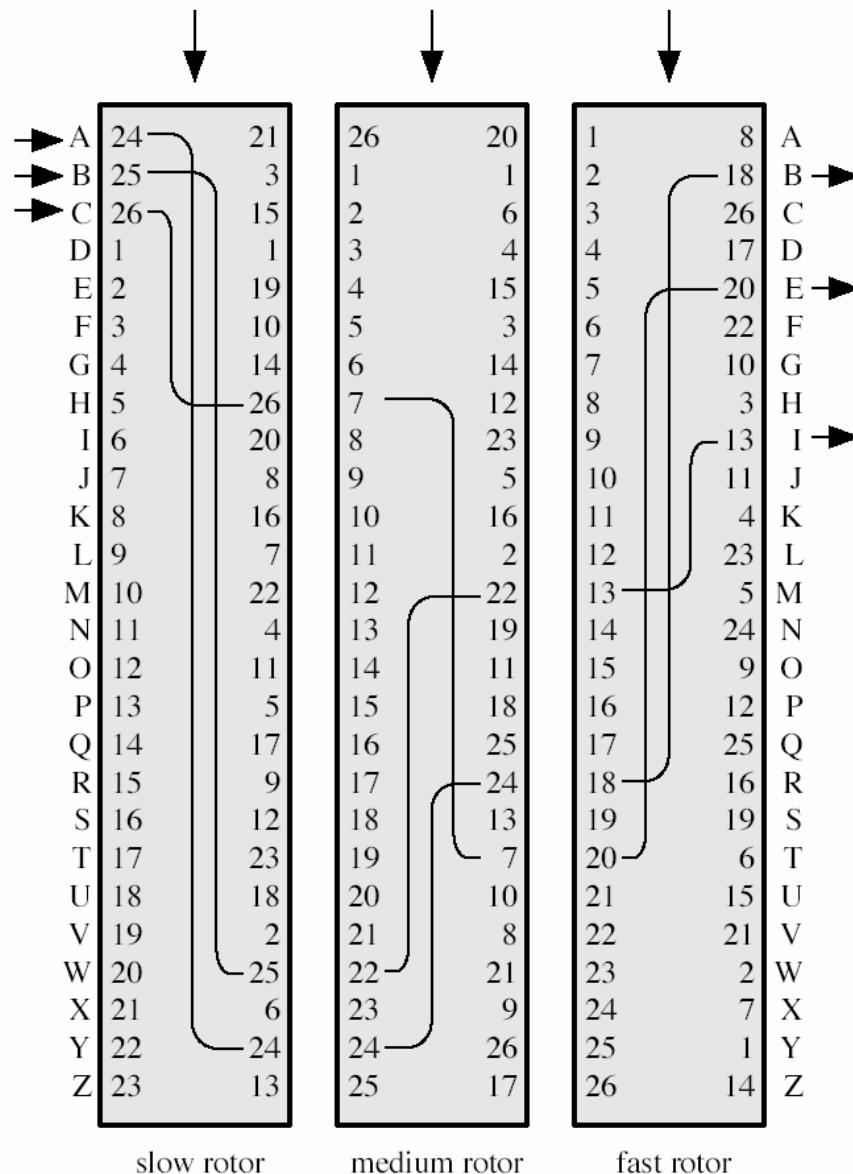
- I cifrari a trasposizione non sono sicuri
- Si possono considerare più cifrari in successione per ottenere un algoritmo maggiormente robusto.
 - Due sostituzioni portano ad una sostituzione più complicata
 - Due permutazioni portano ad una permutazione più complicata
 - **Tuttavia una sostituzione seguita da una trasposizione porta alla creazione di un cifrario molto più resistente**
- I moderni sistemi crittografici sfruttano questo risultato

Macchine a rotazione

- Le macchine a rotazione sono i progenitori dei moderni cifrari
- Ampiamente usate nella II GM
 - German Enigma, Allied Hagelin, Japanese Purple
- Realizzano un cifrario composto a sostituzione alquanto complesso
- Si utilizzava una serie di cilindri, ciascuno dei quai da luogo a una sostituzione, e che erano ruotati dopo la cifratura di ciascuna lettera
- Con 3 cilindri si hanno $26^3=17576$ chiavi

Macchine a rotazione

direction of motion



Steganografia

- È un'alternativa alla crittografia
- Cela un messaggio segreto in uno pubblico
 - Ad esempio usando un sottoinsieme di lettere di un messaggio
 - Utilizzando il LSB in immagini grafiche e file audio e video
- Il principale svantaggio è che c'è bisogno di una grossa mole di dati da inviare per celare dati di dimensione molto più ridotta

Crittografia simmetrica

Algoritmi moderni

Cifrari Simmetrici Contemporanei

- I cifrari simmetrici sono tuttora uno dei tipi di algoritmi crittografici più utilizzati
- Sono utilizzati per implementare i servizi di confidenzialità (segretezza) e autenticazione
- Tra tali cifrari, ci occuperemo nel seguito di DES (Data Encryption Standard)

Cifrari a blocco e a flusso

- I cifrari a blocchi elaborano il messaggio da criptare dividendolo in blocchi di lunghezza fissa; il messaggio cifrato si ottiene elaborando singolarmente ogni blocco
- Possono essere riguardati come cifrari a sostituzione operanti su un alfabeto a cardinalità molto grande
- I cifrari a flusso elaborano il messaggio un bit o un byte per volta
- La maggior parte dei cifrari moderni sono a blocco.

I cifrari a blocco

- In linea di principio, un cifrario a blocco può essere realizzato tramite un look-up table
- Infatti, un cifrario a blocco non è altro che un cifrario a sostituzione
- Utilizzando blocchi di n bit, tale tabella richiede una memoria pari a $n2^n$. Questa è di fatto la chiave del codice
- Per $n=64$ è $64 \times 2^{64} = 10^{21}$ bit.
- Per ridurre l'ingombro di memoria, si utilizzano cifrari “strutturati”
- In pratica, cifrari complicati sono ottenuti componendo cifrari più semplici

Cifrari basati su Sostituzioni e Permutazioni (by Claude Shannon)

- Il lavoro “Communication Theory of Secrecy Systems,” apparso nel 1949, contiene i concetti fondamentali utilizzati nei cifrari a blocco moderni
- In realtà, tali contenuti erano già presenti nel report “A mathematical theory of cryptography,” datato 01/09/1946, e che fu tenuto segreto
- Shannon introdusse l’idea di utilizzare reti costituite da cifrari più semplici, operando le operazioni di sostituzione e permutazione

Cifrari basati su Sostituzioni e Permutazioni (by Claude Shannon)

- Le reti S-P sono alla base dei moderni cifrari a blocco
- Tali reti sono basate su due semplici operazioni crittografiche che abbiamo già esaminato:
 - sostituzione (S-box)
 - permutazione (P-box)
- Tali reti introducono sul messaggio gli effetti di **confusione** e **diffusione**

Confusione e Diffusione

- Un buon cifrario dovrebbe mascherare completamente le proprietà statistiche del plaintext (esempio: one-time pad)
- Per ottenere in pratica un cifrario “quasi-ideale” Shannon suggerì di utilizzare strutture che portassero alla
- **diffusione** – spalma la struttura statistica del plaintext su blocchi di testo cifrato
- **confusione** – rende la relazione tra il testo cifrato e la chiave utilizzata il più complicata possibile

Confusione e Diffusione

- La confusione si realizza mediante operazioni di
 - *Permutazione*: nasconde le frequenze dei q -grammi, con $q > 1$
 - *Combinazione*: ogni bit del ciphertext viene fatto dipendere da molti bit del plaintext, in modo da nascondere le frequenze delle singole lettere (Idealmente, ogni lettera del testo cifrato dovrebbe dipendere da tutti i caratteri del testo in chiaro)

Confusione e Diffusione

Esempio:

$$X = (m_1, \dots, m_k) \quad (m_i \in \mathbb{Z}_{26});$$

$$Y_i = \sum_{j=1}^k m_{i+j} \pmod{26}.$$

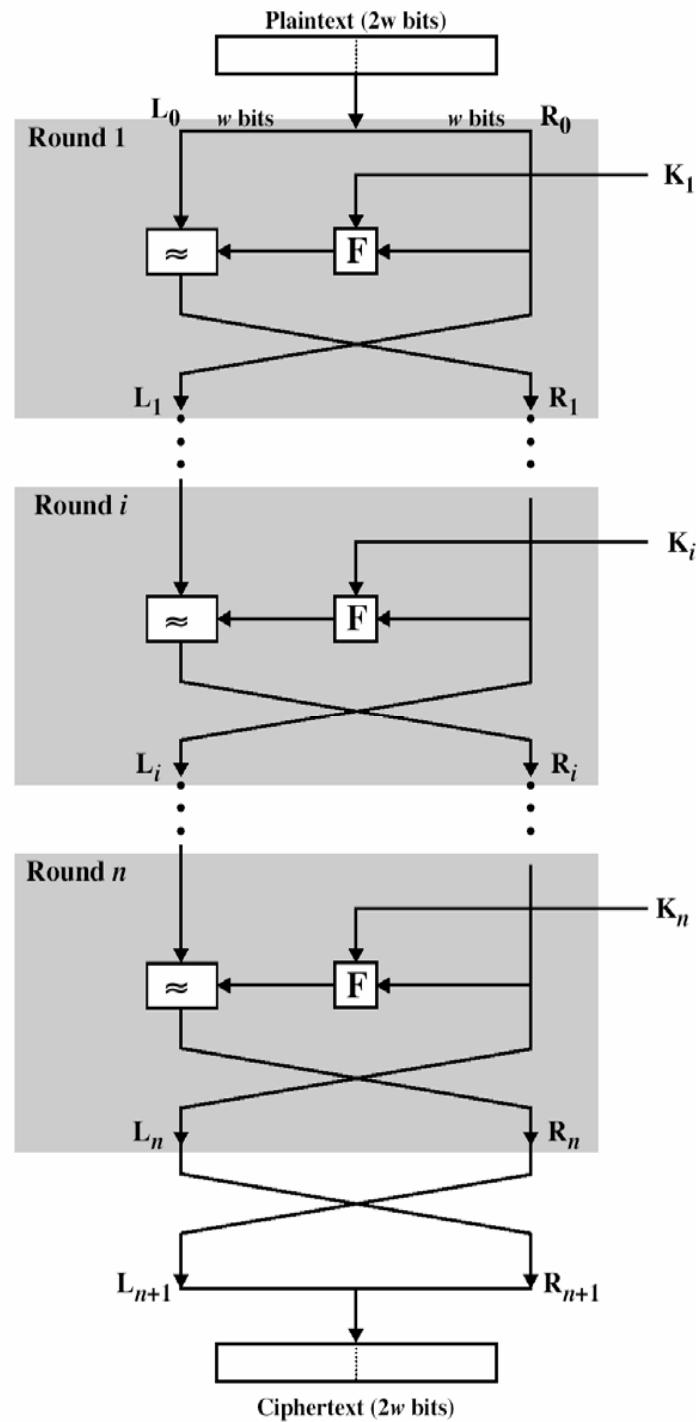
In questo caso ogni lettera del ciphertext dipende da ogni lettera del plaintext.

La “Struttura Feistel”

- Primi anni 70: Horst Feistel, ricercatore presso il Thomas J. Watson Research Lab dell'IBM inventò una struttura ad-hoc facilmente invertibile e molto semplice da realizzare
 - Era basata sull'uso di cifrari composti
- Il blocco da cifrare è diviso in due parti
- Utilizza le funzioni di permutazione e sostituzione indicate da Shannon

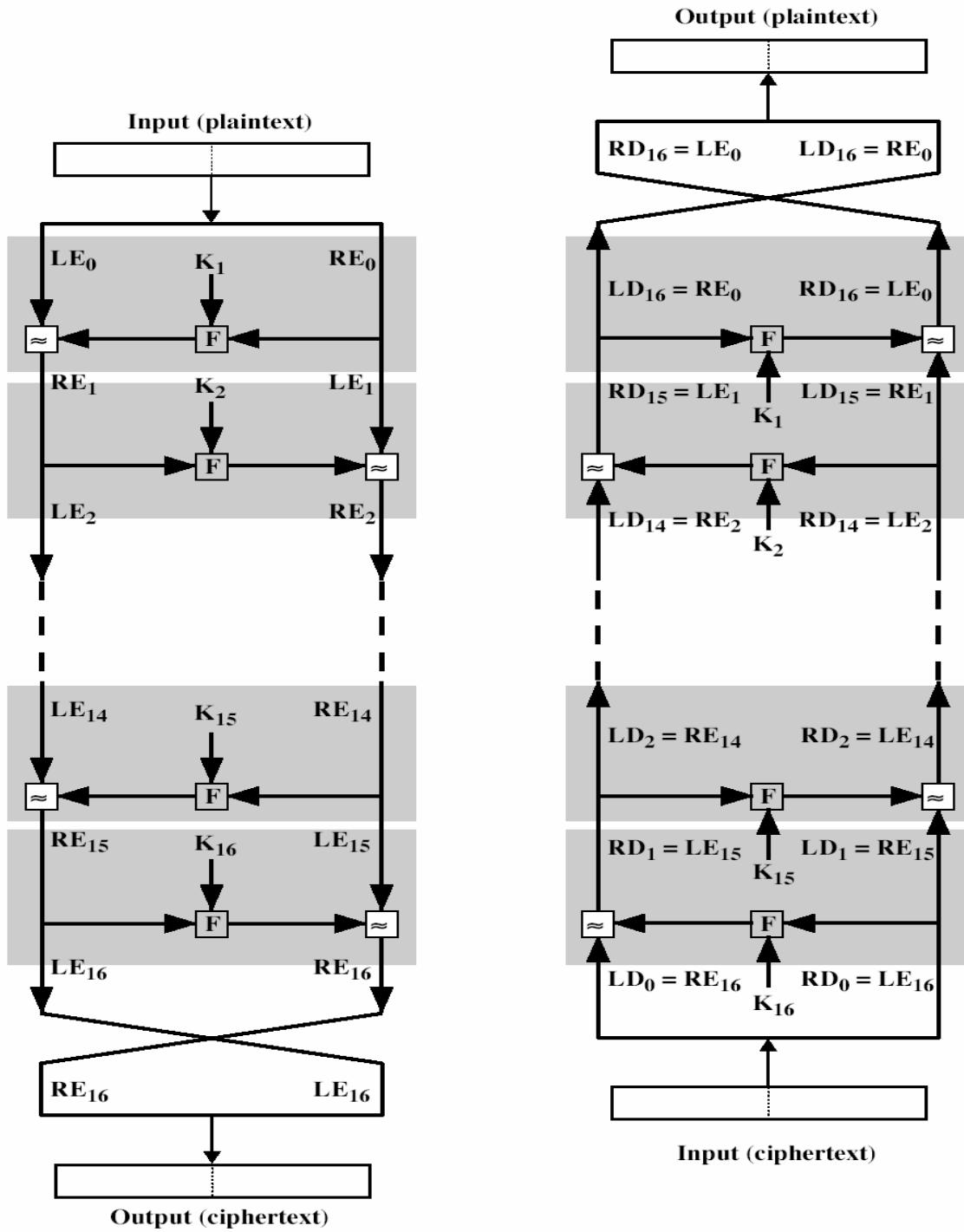
La “Struttura Feistel”

- Forma particolare della struttura proposta da Shannon
- Implementa n volte la stessa funzione
- Elabora separatamente parte dx e parte sx
- In ogni round solo metà dei dati è alterata



Parametri di Progetto della Struttura Feistel

- **Dimensione del blocco**
 - Ha impatto sulla sicurezza, ma anche sulla velocità di esecuzione della criptazione e decriptazione
- **Dimensione della chiave**
 - Idem, come sopra
- **Numero di round (iterazioni)**
 - Idem, come sopra
- **Generazione delle sottochiavi**
 - Funzioni complesse rendono l'analisi crittografica più complicata, ma rallentano il codice
- **Funzione round**
 - Idem, come sopra
- **Velocità del software e facilità di analisi**
 - È bene che l'algoritmo sia implementabile con software veloci e, anche, che sia semplice, in modo che l'analisi crittografica possa porne in evidenza eventuali punti deboli.



Criptazione e Decrittazione del Cifrario di Feistel

Non è richiesto che la F sia invertibile!!

Data Encryption Standard (DES)

- E' uno dei cifrari a blocco più ampiamente utilizzati nel mondo
- Adottato nel 1977 dal NBS (now NIST, National Institute of Standards and Technology)
 - come FIPS PUB 46 (Federal Information Processing Standard)
- Cripta parole di 64-bit usando una chiave di 56 bit
- Vi sono state moltissime discussioni circa la sua sicurezza

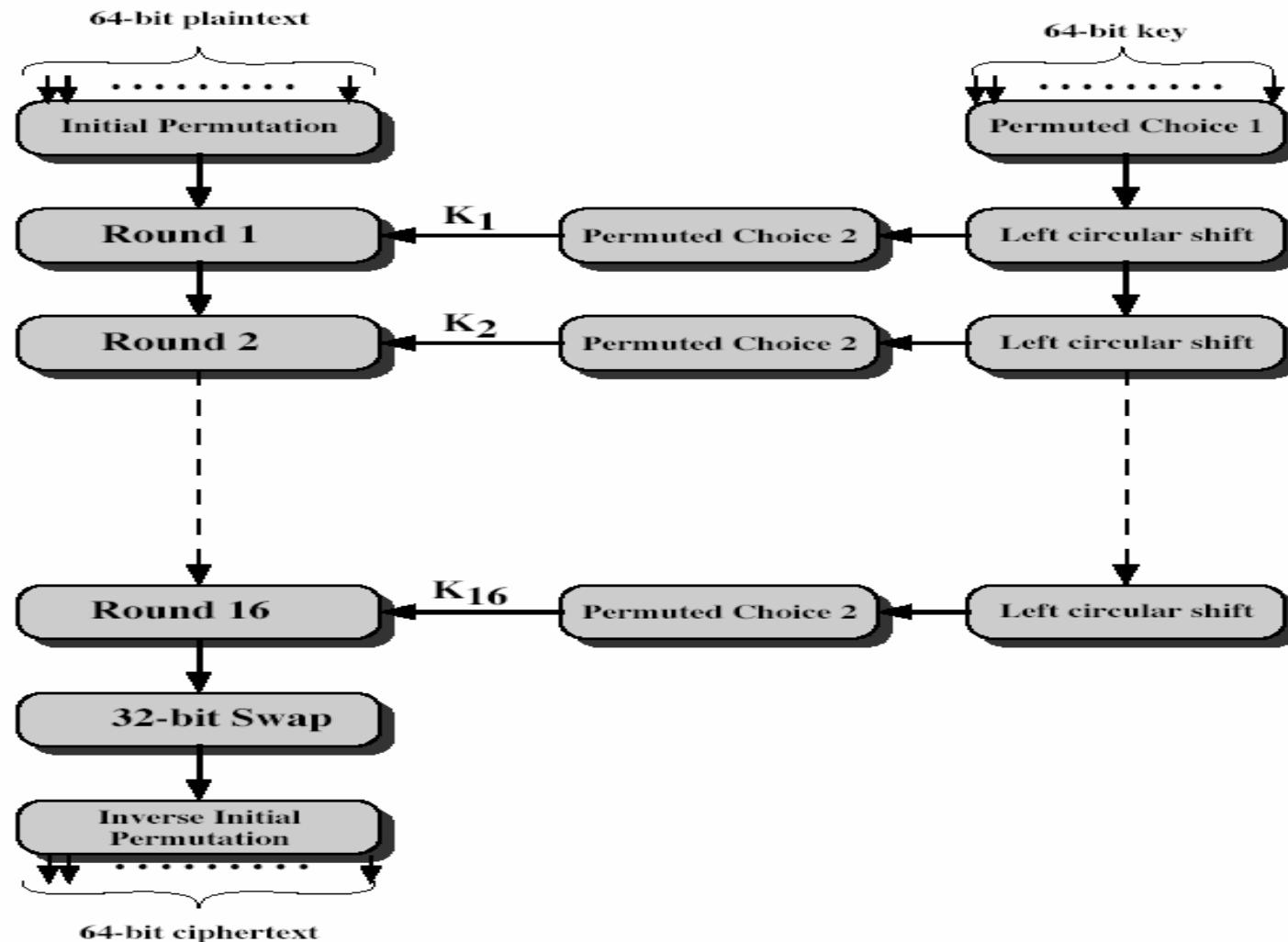
La storia di DES

- L' IBM sviluppò nel 1971 il cifrario LUCIFER
 - Il team di sviluppo era guidato da Feistel
 - Criptava parole di 64 bit con una chiave di 128 bit
 - Fu venduto ai Lloyd di Londra per l'utilizzo nei cash dispenser
- A partire da LUCIFER, l'IBM decise di svilupparne una versione più evoluta che potesse essere realizzata e commercializzata su un chip
- Nel 1973 NBS emanò un call for proposal per uno standard di cifratura nazionale
- IBM sottopose la versione evoluta di LUCIFER che fu poi accettata e divenne DES

Le polemiche su DES

- DES è un algoritmo pubblico
- Vi sono state polemiche in quanto
 - La chiave è di 56 bit (vs Lucifer 128-bit)
 - I criteri di progetto dei round non sono noti
- Le analisi che susseguirono hanno tuttavia mostrato che DES è un buon algoritmo
- DES è stato ampiamente utilizzato, specialmente per transazioni finanziarie

Criptazione DES



La Permutazione Iniziale

- È il primo stadio dell'algoritmo
- Riordina i dati in ingresso
- I bit pari sono posti nella metà sinistra, quelli dispari nella metà destra
- La struttura è alquanto regolare, cosicchè possa essere implementata in modo semplice in hardware

La Permutazione Iniziale

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Esempio: L'output di IP sarà b(58) b(50) b(42) b(34) b(15) b(7)

Struttura dei round in DES

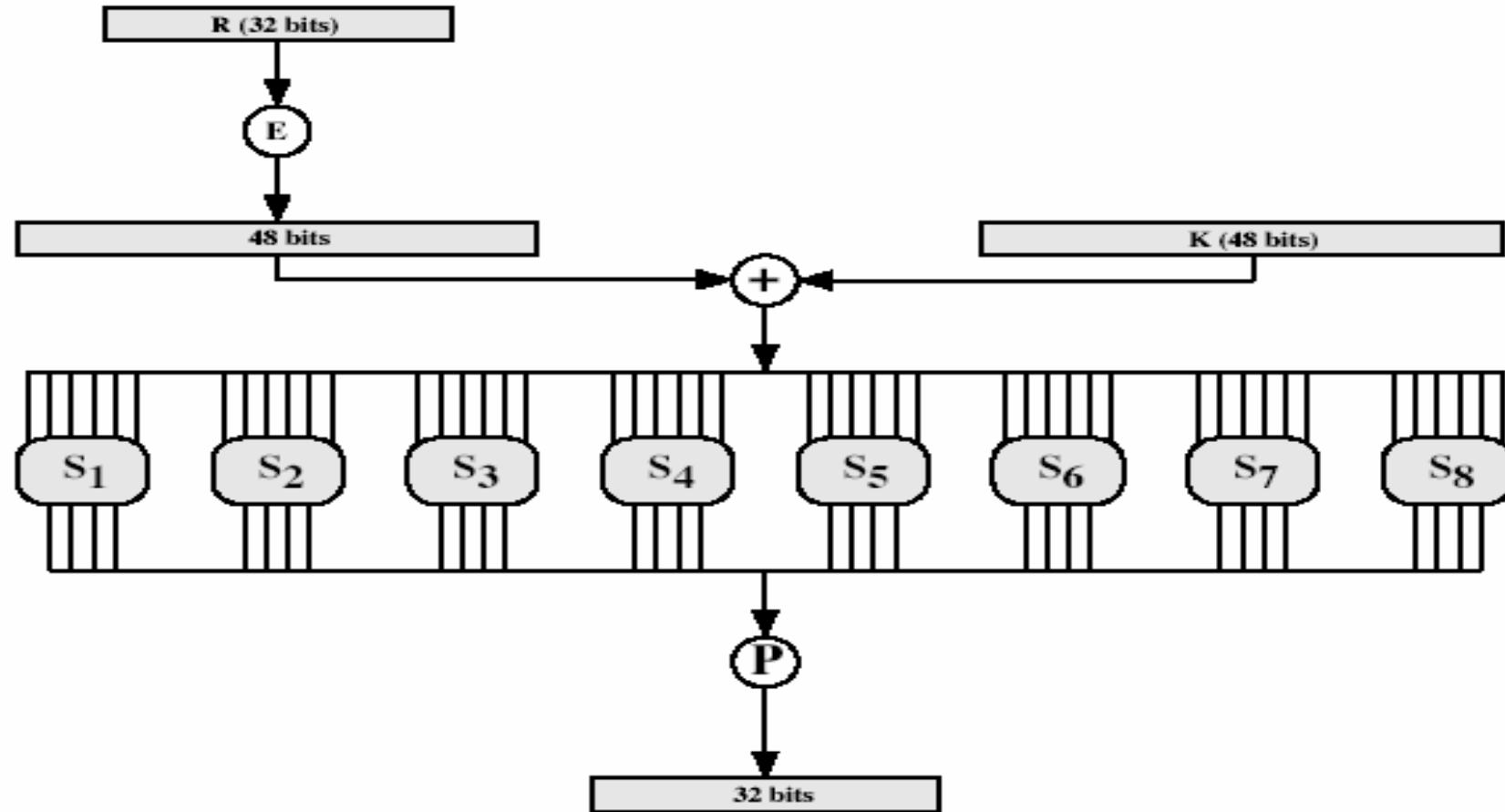
- Divide i dati nella parte sinistra (L) e destra (R)
- Come nella classica struttura Feistel, si ha:

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

$$R_i = L_{i-1} \text{ xor } F(R_{i-1}, K_i)$$

- La funzione F elabora i 32 bit della parte R e la sottochiave di 48 bit come segue:
 - espande R a 48 bit mediante una permutazione/espansione (vedi lucido seguente)
 - Somma il risultato alla sottochiave
 - Invia il tutto a 8 S-boxes per avere un output di 32 bit
 - I 32 bit sono assoggettati a una permutazione finale

Struttura dei round in DES



L'Espansione/Permutazione (da 32 a 48 bit)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

La Permutazione Finale

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Le S-Box

- Vi sono 8 S-Box che sono funzioni che accettano in ingresso 6 bit e ne producono 4
- Ciascuna S-box è una matrice 4×16 contenente interi in $\{0, 1, \dots, 15\}$
 - I bit 1 & 6 selezionano la riga
 - I bit 2-5 selezionano la colonna
 - Il risultato è l'espansione binaria dell'elemento selezionato della matrice
- L'output delle S-box dipende sia dai dati che dalla chiave

Le S-Box in DES

S₁	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S₂	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S₃	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S₄	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Le S-Box in DES

S₅	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S₆	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S₇	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S₈	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Generazione delle sottochiavi

- In ogni round sono utilizzate delle sottochiavi
- La chiave originaria è lunga 64 bit, ma un bit ogni 8 non conta, quindi si tratta solo di 56 bit
 - Vi è una permutazione iniziale della chiave, che viene poi suddivisa in due metà di 28 bit ciascuna
 - Vi sono poi 16 stadi in cui:
 - Si selezionano 24 bit da ciascuna metà
 - Si effettua una permutazione (PC2),
 - Si ruota verso sinistra ciascuna metà di un numero di posizioni pari a 1 o a 2

La chiave in ingresso

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

La prima permutazione

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

La seconda permutazione e gli shift ciclici

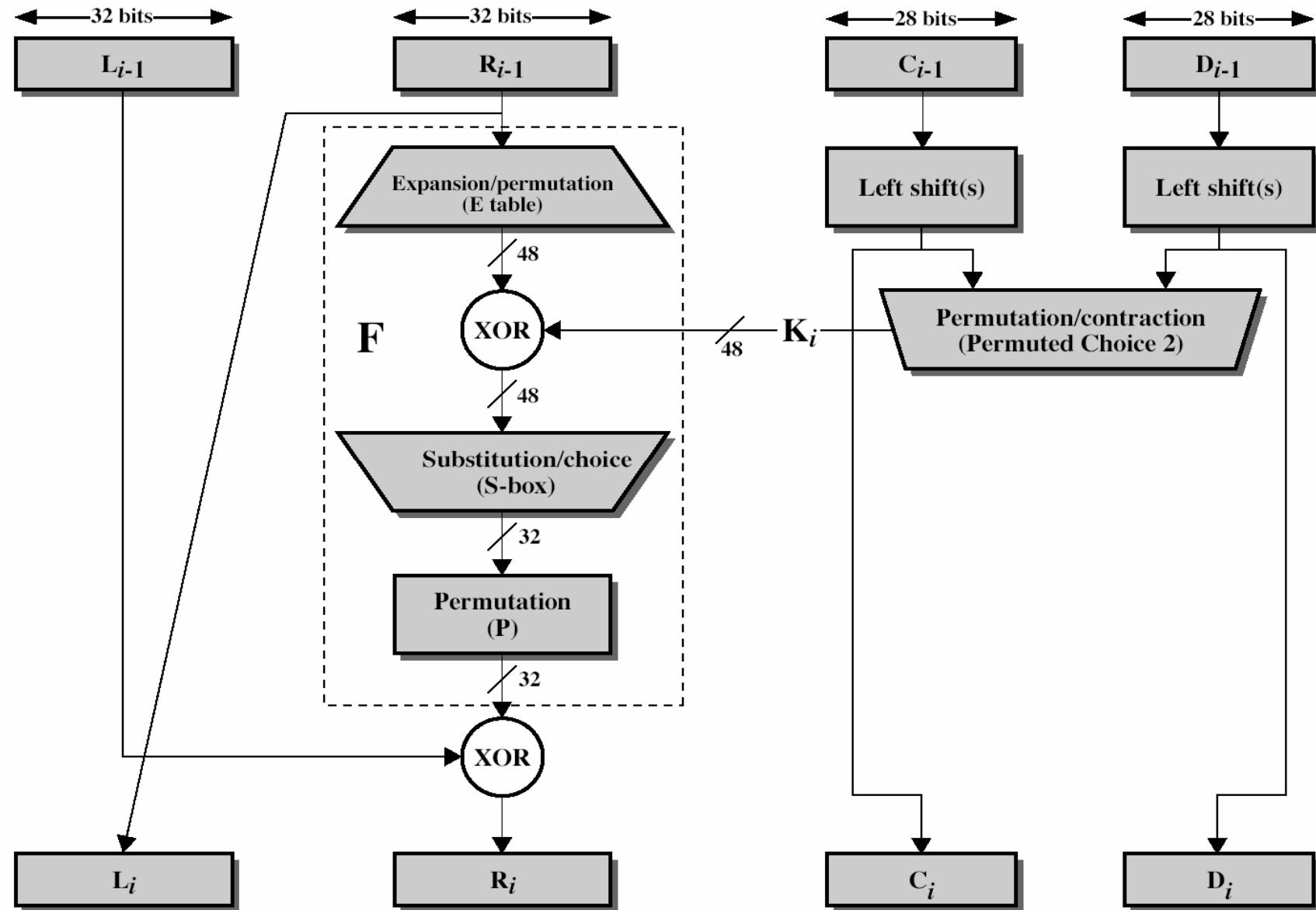
(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Un singolo round in DES



La decriptazione in DES

- La decriptazione è analoga alla criptazione
- In virtù della struttura Feistel, la decriptazione è analoga alla criptazione con le sottochiavi utilizzate in ordine inverso
- Nota che la permutazione finale è l'inverso della permutazione iniziale

L'effetto valanga

- E' una caratteristica desiderata per ogni algoritmo di criptazione
- Un cambiamento di pochi bit nel plaintext deve provocare un cambiamento di quanti più bit nel ciphertext
- DES possiede un forte effetto valanga

L'effetto valanga in DES

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

La forza di DES: la chiave

- Vi sono $2^{56} = 7.2 \times 10^{16}$ possibili chiavi
- Nel 1977 l'attacco a forza bruta non era fattibile
- Le cose sono cambiate negli ultimi anni
 - nel 1998 fu rotto in 3 giorni dalla EFF (Electronic Frontier Foundation) con una macchina dedicata del costo di \$ 250.000
 - nel 1999 tale tempo fu ridotto a 22 ore!
- Al giorno d'oggi, vi sono alternative a DES, che vedremo più avanti nel corso (es. AES, 3DES)

Criteri di progetto di DES

- I criteri di progetto si rifanno al progetto LUCIFER di Feistel
- Numero di round
 - La loro numerosità rafforza il codice
- La funzione F:
 - È quella che introduce la “confusione”, **deve** essere non lineare (robustezza all’attacco known plaintext), e possedere l’effetto valanga
- Gestione della chiave
 - La creazione della sottochiave deve essere complicata; deve essere garantito l’effetto valanga

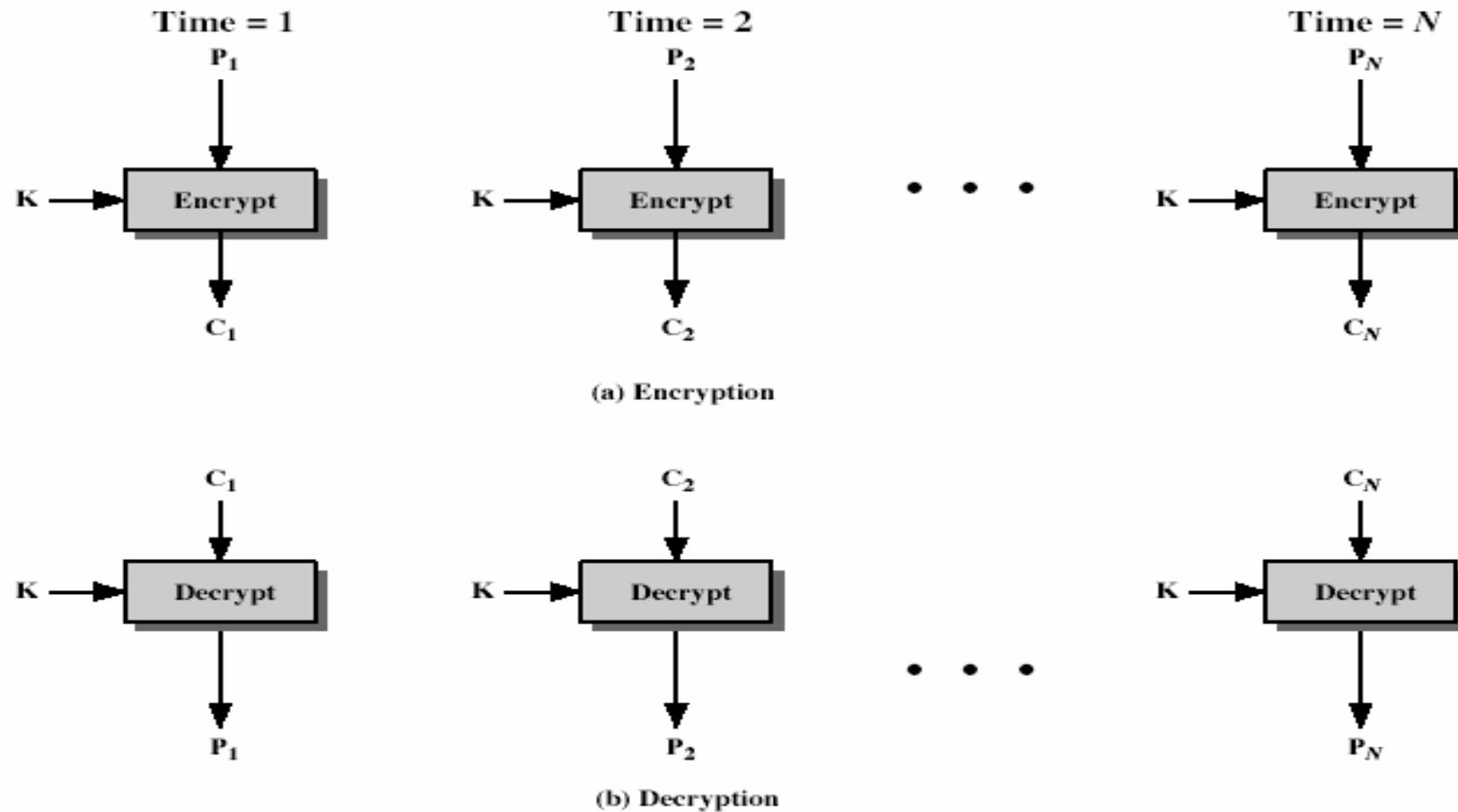
Modalità di utilizzo dei cifrari a blocco

- I Cifrari a blocchi criptano parole di lunghezza fissa (esempio: DES elabora blocchi di 64 bit)
- Nella realtà, con grosse moli di dati, vi sono molteplici blocchi da criptare
- L' ANSI standard **ANSI X3.106-1983 Modes of Use** ha stabilito 4 possibili modi di utilizzo dei cifrari a blocco
- I modi di utilizzo usualmente considerati sono 5

Electronic Codebook (ECB)

- Il messaggio è suddiviso in blocchi indipendenti che sono poi criptati singolarmente
- Si tratta a tutti gli effetti di un cifrario a sostituzione
- È la modalità ideale per la trasmissione di brevi quantità di dati

Electronic Codebook (ECB)



Svantaggi della modalità ECB

- La ridondanza del plaintext viene riportata nel ciphertext
 - Blocchi uguali di plaintext in tempi diversi produrranno blocchi analoghi di ciphertext
 - Se il messaggio è strutturato (header comune) si possono ottenere copie di testo in chiaro/testo cifrato su cui lavorare
- La modalità ECB viene utilizzata per l'invio di piccole quantità di dati

Cipher Block Chaining (CBC)

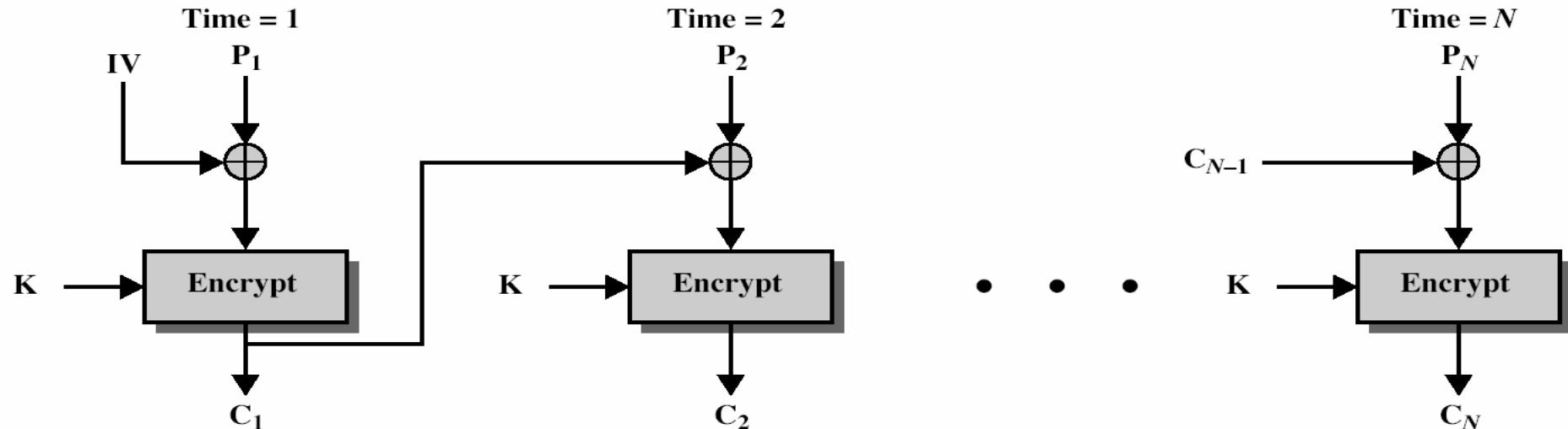
- Il plaintext è diviso in parole di 64 bit
- Tali parole sono però collegate tra loro durante la criptazione
- Ciascun blocco cifrato è concatenato (tramite uno XOR) col successivo blocco di plaintext
- Il primo blocco di plaintext fa uso di un vettore di inizializzazione (IV):

$$C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$$

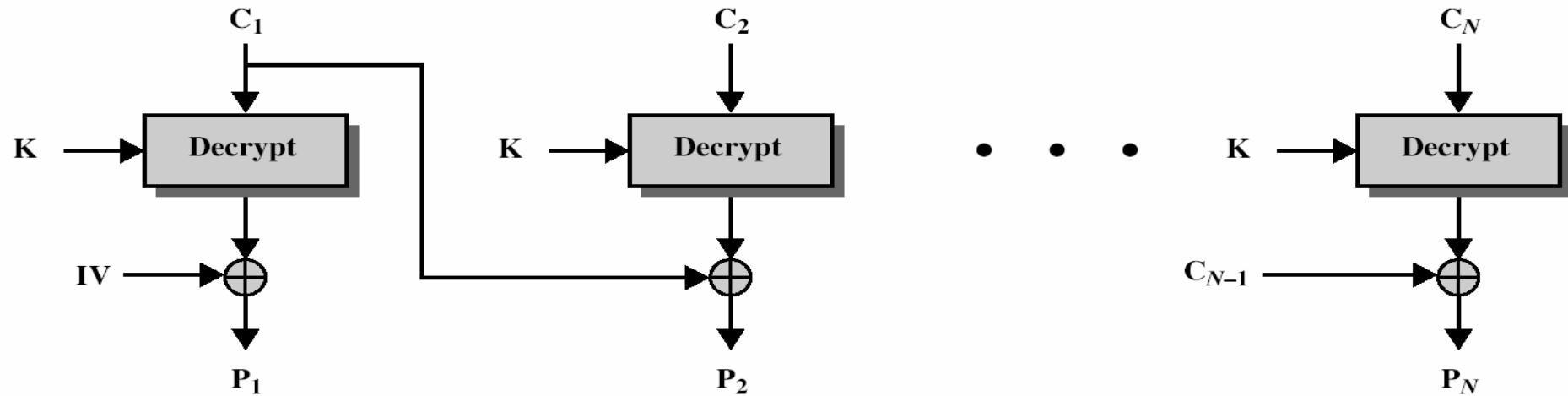
$$C_{-1} = IV$$

- La modalità CBC viene usata per la criptazione di grosse quantità di dati

Cipher Block Chaining (CBC)



(a) Encryption



(b) Decryption

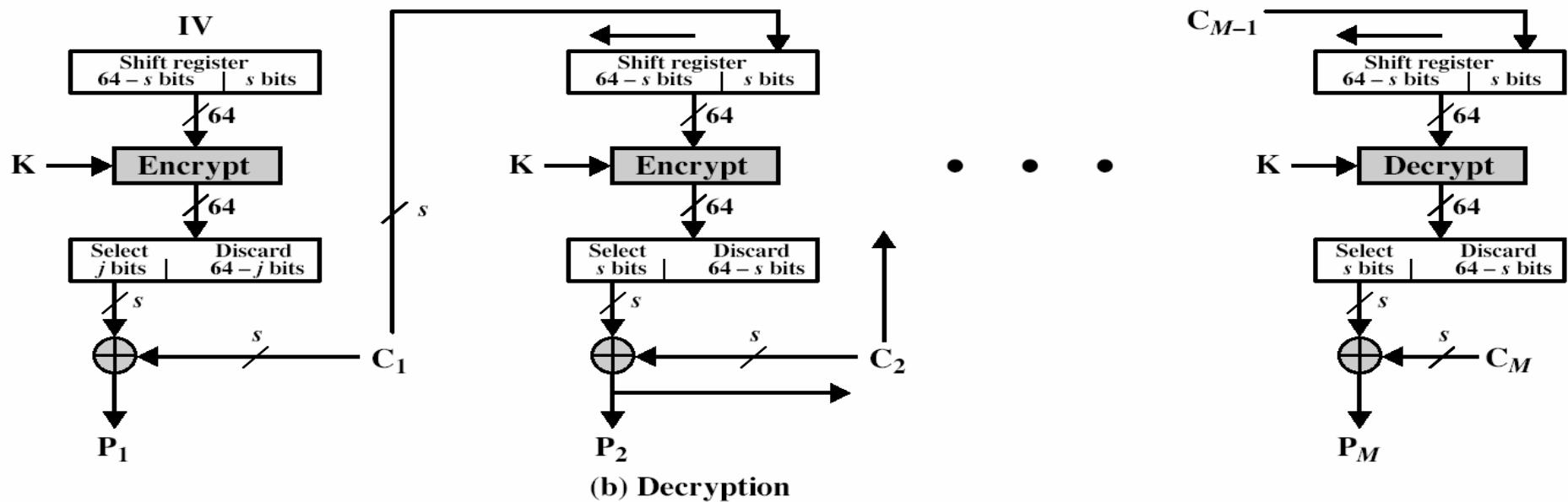
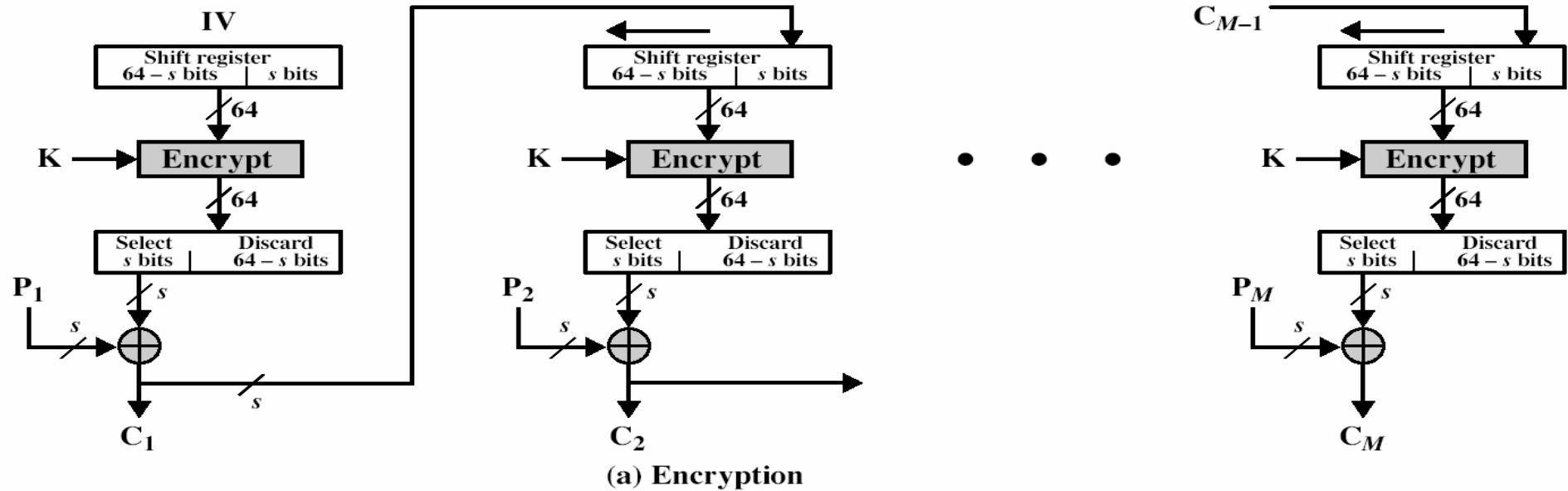
Modalità CBC: vantaggi e svantaggi

- Ciascun blocco di ciphertext dipende da **tutti i precedenti** blocchi di plaintext
- Un cambiamento in un singolo blocco ha effetto su tutti i blocchi cifrati seguenti
- C'è bisogno di un vettore di inizializzazione (IV) noto al trasmettitore e al ricevitore
 - IV tuttavia non può essere inviato in chiaro
 - IV deve essere inviato in forma criptata (ad esempio con modalità ECB) o deve assumere un valore fisso e noto

Cipher FeedBack (CFB)

- Il messaggio è considerato come un flusso di bit
- Viene aggiunto all'uscita del testo cifrato
- Il risultato è poi utilizzato allo stadio successivo
- Lo standard stabilisce che possano essere criptati un numero arbitrario di bit per volta (ovviamente non superiore a 64)
- usi: criptazione di dati in tempo reale

Cipher FeedBack (CFB)



CFB: Vantaggi e Svantaggi

- È appropriata quando i dati arrivano in bytes o gruppi di bit
- È il cifrario di flusso più comune
- Lo svantaggio principale è la sensibilità ad errori sul canale di trasmissione: Un errore si propagherà anche nei blocchi successivi

Output FeedBack (OFB)

- Il messaggio è considerato come un flusso di bit
- Viene aggiunto all'uscita del testo cifrato
- Il risultato è poi utilizzato allo stadio successivo
- L'informazione di feedback non dipende dal plaintext, ma solo dalla chiave
- Può quindi essere calcolata preventivamente

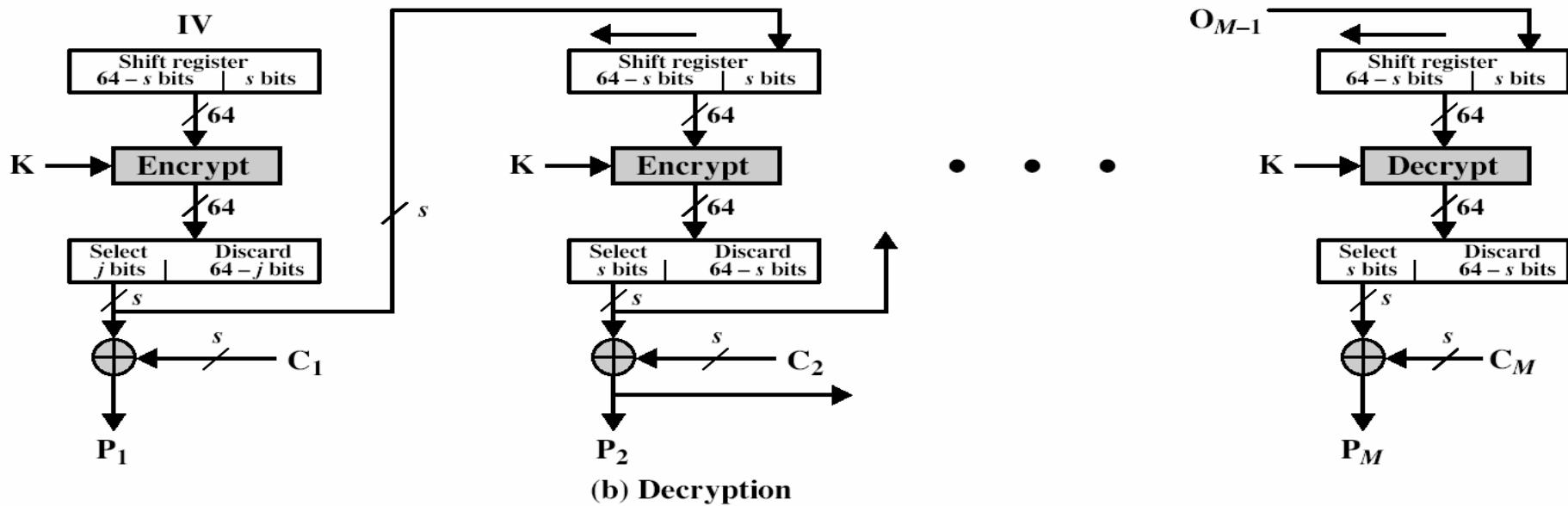
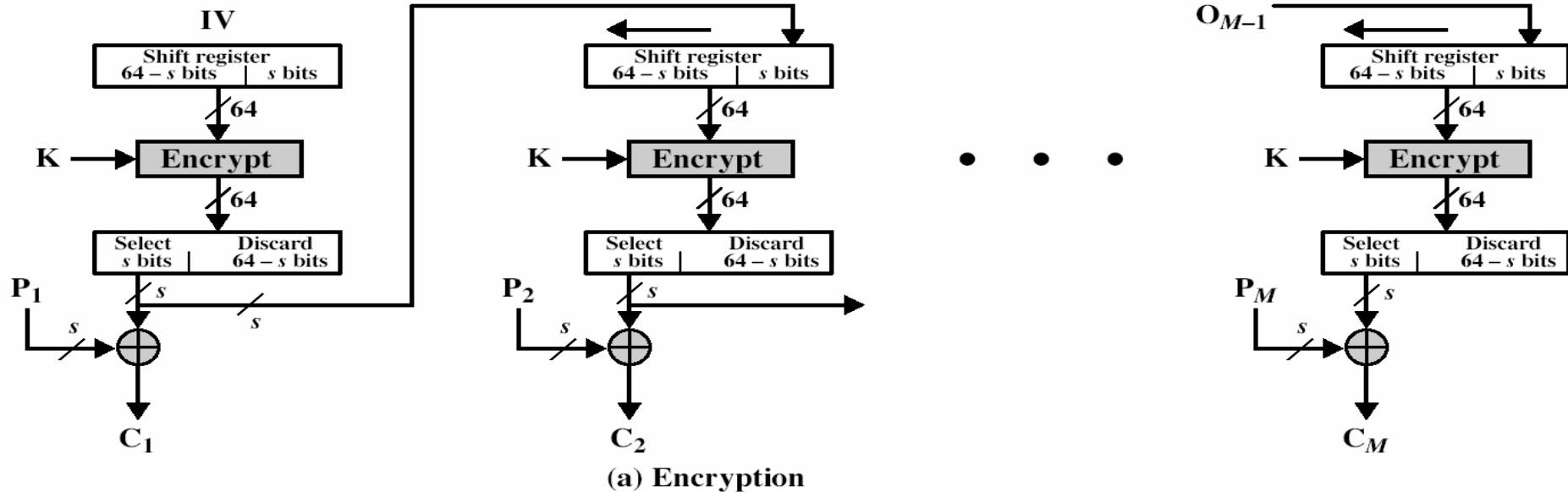
$$C_i = P_i \text{ XOR } O_i$$

$$O_i = \text{DES}_{K1}(O_{i-1})$$

$$O_{-1} = \text{IV}$$

- Usi: criptazione di flussi per trasmissione su canali rumorosi

Output FeedBack (OFB)



OFB: Vantaggi e Svantaggi

- Utilizzato quando possono esservi errori nella trasmissione
- A prima vista è simile al CFB
- In realtà il feedback è indipendente dal messaggio
- Il feedback fa sì che si utilizzi una chiave tempo-variante
- Trasmettitore e ricevitore devono quindi operare in sincronia
- È vulnerabile a un attacco a modifica del flusso: cambiando un bit del ciphertex si ottiene il cambiamento dello stesso bit nel plaintext

Counter (CTR)

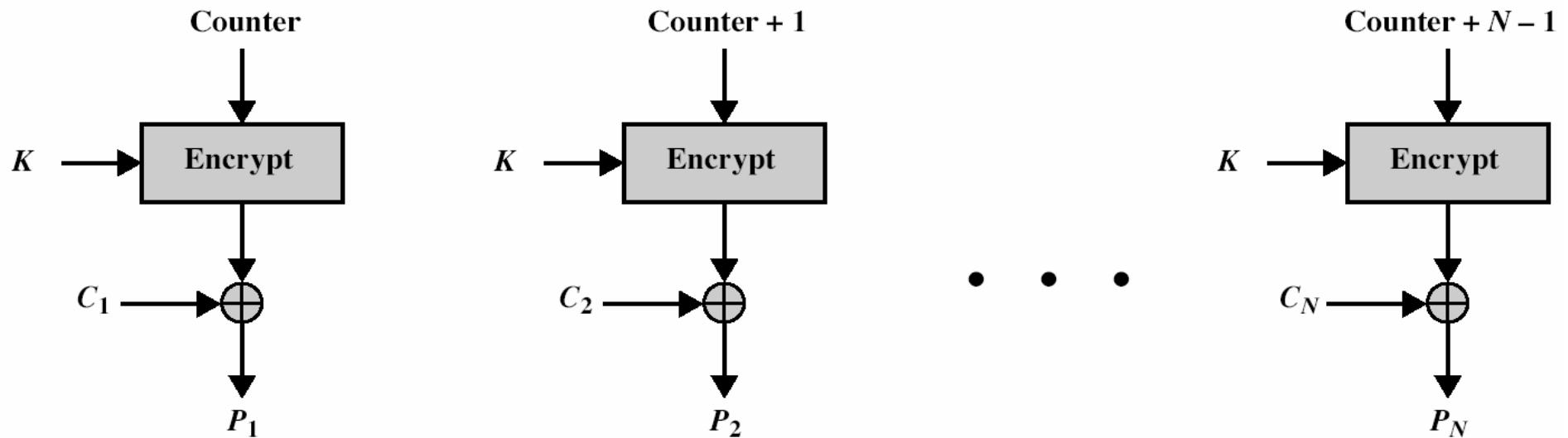
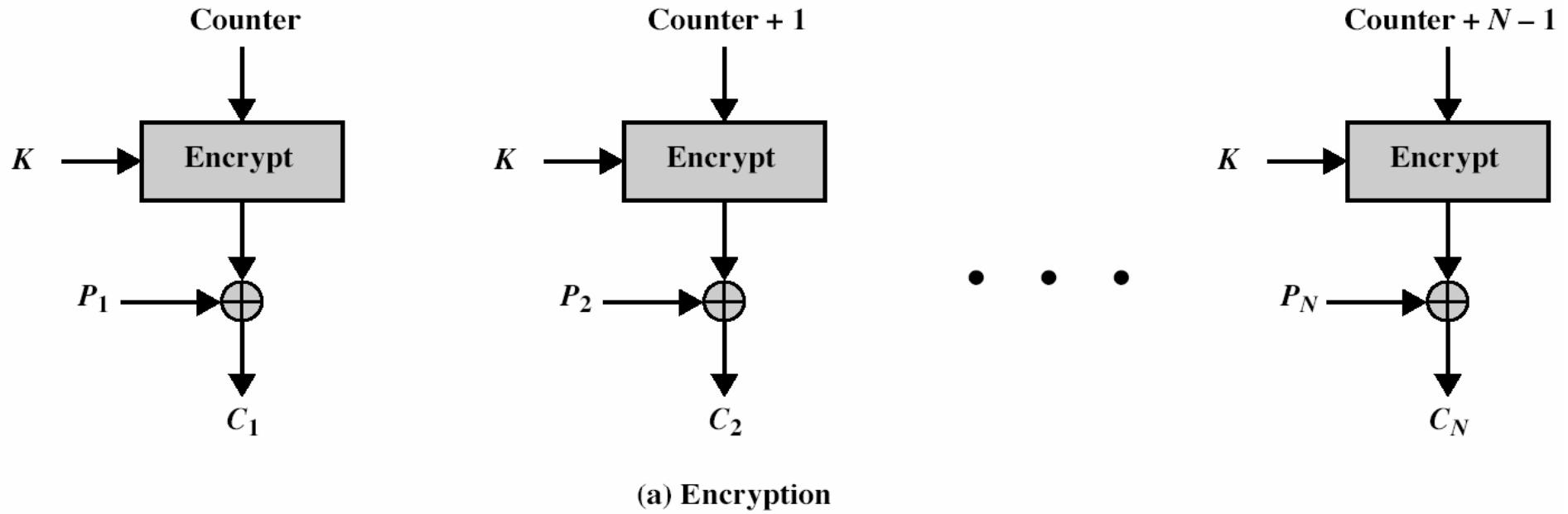
- È una modalità proposta nel 1979, ma che ha acquisito importanza solo in epoca recente
- È simile a OFB, con la differenza che viene criptato un contatore invece che un'informazione inviata in feedback
- Il contatore deve essere diverso per ogni blocco di plaintext

$$C_i = P_i \text{ XOR } O_i$$

$$O_i = \text{DES}_{K1}(i)$$

- Uso: criptazione nelle reti ad alta velocità (es. ATM)

Counter (CTR)



CTR: Vantaggi e Svantaggi

- efficienza
 - Si presta all'implementazione parallela di più criptazioni
 - Se si dispone di memoria, si possono calcolare le uscite O_i prima che il plaintext sia disponibile
 - È indicato per reti veloci e traffico a burst
- I blocchi di dati possono essere criptati/decriptati in modo casuale (random access)
- È una modalità sicura almeno quanto le altre
- Si deve garantire però che non venga usata lo stesso counter con la stessa chiave più di una volta

Estensioni di DES

Limiti di DES

- DES è vulnerabile ad attacchi a forza bruta (lunghezza della chiave: 56 bit):
 - nel 1977 si riteneva che già esistesse la tecnologia per violare DES in 10 ore (costo 20 milioni USD)
 - nel 1998 la *Electronic Frontier Foundation* costruisce una macchina ‘DES cracker’ in grado di violare DES in tre giorni (costo 250 000 USD)

NOTA: l'attacco a forza bruta richiede che l'analista sia in grado di riconoscere il testo in chiaro decifrato!

Soluzioni

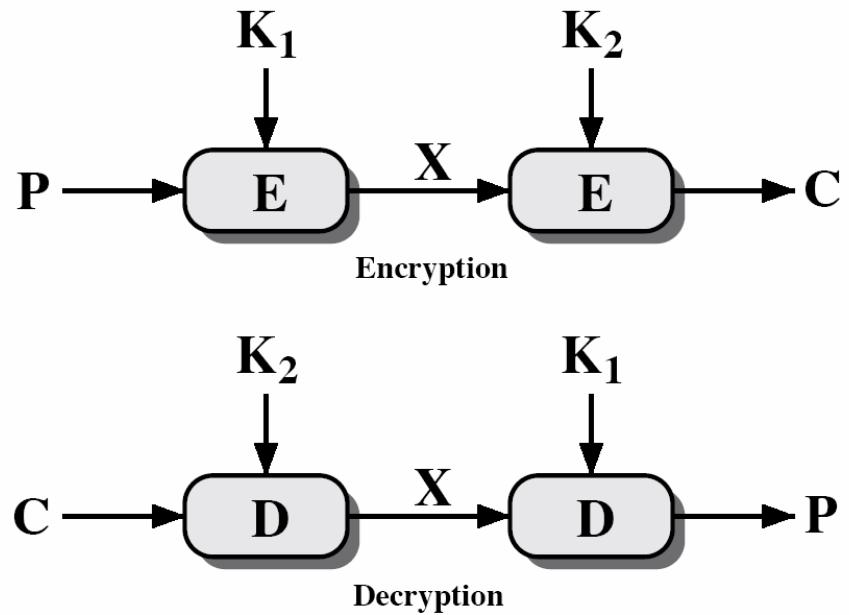
- Progettare un algoritmo di crittografia completamente nuovo e più resistente agli attacchi a forza bruta (i.e., AES).
- Utilizzare una crittografia DES multipla con più chiavi:
 - Double DES
 - Triple DES a due chiavi
 - Triple DES a tre chiavi

Double DES

- Due fasi di crittografia in cascata che utilizzano due chiavi distinte K_1 e K_2 :

$$C = E_{K_2}[E_{K_1}[P]]$$

$$P = E_{K_1}[E_{K_2}[C]]$$



Double DES

- Apparentemente Double DES usa una chiave di $56 \times 2 = 112$ bit.
- Domanda: date due chiavi K_1 e K_2 , è possibile trovare una chiave K_3 tale che

$$E_{K_2}[E_{K_1}[P]] = E_{K_3}[P] ?$$

- Risposta: in generale, applicando DES due volte si ottiene un mappaggio che non può essere ottenuto da una sola applicazione di DES.

Attacco Meet-in-the-Middle

$$C = E_{K_2}[E_{K_1}[P]] \Rightarrow X = E_{K_1}[P] = D_{K_2}[C]$$

- Data una coppia nota (P, C) , l'attacco procede come segue:
 - Si esegue la crittografia di P per tutti i 2^{56} valori di K_1
 - Si esegue la decrittografia di C per tutti i 2^{56} valori di K_2
 - Quando si trova una corrispondenza, si esegue il test delle due chiavi risultanti contro una nuova coppia (P_a, C_a) . Se le due chiavi producono il testo cifrato corretto, si tratta delle chiavi corrette.

Attacco Meet-in-the-Middle

- Per ogni testo in chiaro P , Double DES produce uno fra 2^{64} possibili valori cifrati.
- Poiché Double DES usa chiavi di 112 bit, ci sono in media $2^{112}/2^{64} = 2^{48}$ chiavi che producono lo stesso testo cifrato C .
- L'attacco meet-in-the-middle produce in media 2^{48} falsi allarmi sulla coppia (P,C) .
- Utilizzando una seconda coppia (P_a, C_a) , i falsi allarmi si riducono a $2^{48-64} = 2^{-16}$, ovvero la probabilità di individuare le chiavi corrette è $1 - 2^{-16}$.

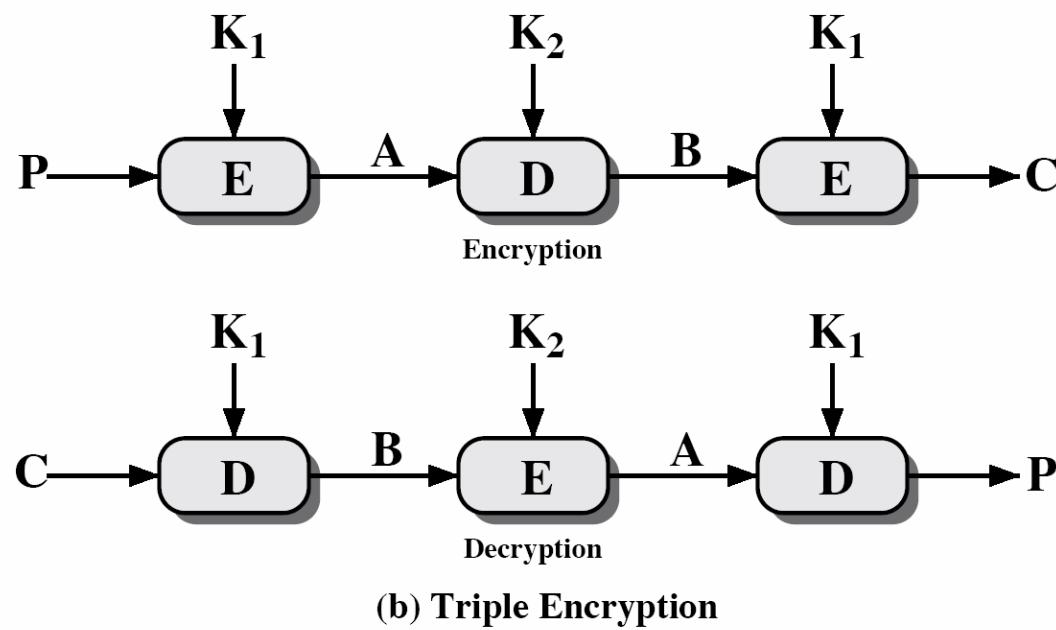
Attacco Meet-in-the-Middle

- L'attacco meet-in-the-middle ha una complessità computazionale pari 2^{56} .
- Di conseguenza, pur utilizzando chiavi di 112 bit, Double DES può essere violato in maniera relativamente semplice con un attacco a testo in chiaro noto.

NOTA: Si ricordi che DES può essere violato con un attacco a testo in chiaro noto con un impegno pari a 2^{55} .

Triple DES con due chiavi

- Una contromisura contro l'attacco meet-in-the-middle è l'uso di tre fasi di crittografia:



Triple DES con due chiavi

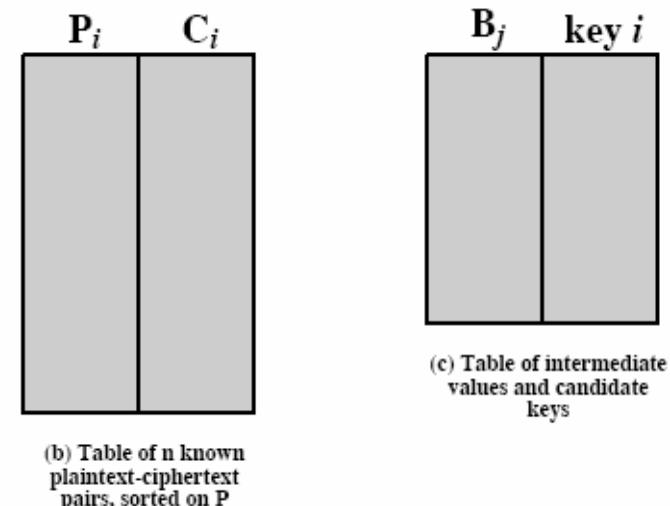
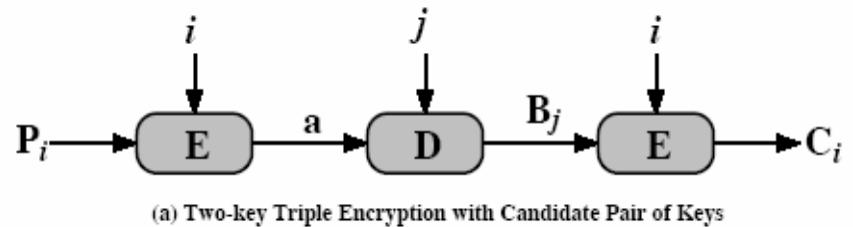
- Vengono ancora utilizzate due chiavi crittografiche da 56 bit ciascuna.
- La presenza di una decrittografia nella seconda fase consente agli utenti Triple DES di decrittografare i dati crittografati dagli utenti DES (basta porre $K_1=K_2$).
- L'attacco meet-in-the-middle a testo in chiaro noto avrebbe una complessità 2^{112} .

Attacchi a Triple DES con due chiavi

- Se è nota la coppia (A,C), il problema si riduce ad un attacco a Double DES.
- In pratica, se le due chiavi K_1 e K_2 sono sconosciute, A non è noto all'analista. In tal caso si procede fissando una valore potenziale di A e si tenta di trovare una coppia nota (P,C) che produce A.

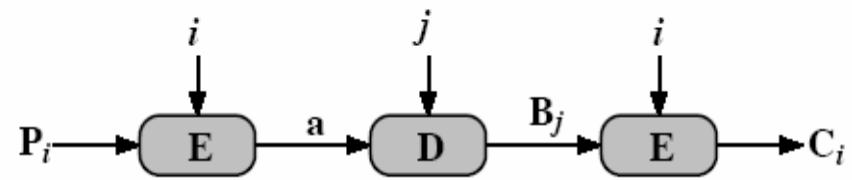
Schema di attacco (1/3)

- Ottenere n coppie (P, C) e inserirle nella tabella (b) ordinate per valori di P .
- Scegliere un valore $A=a$. Per tutte le 2^{56} chiavi $K_i=i$ calcolare $P_i=D_i[a]$. Per ciascun P_i presente anche nella tabella (b) creare una voce nella tabella (c) contenente K_i e $B_i=D_i[C]$. Ordinare la tabella (c) sulla base dei valori di B .

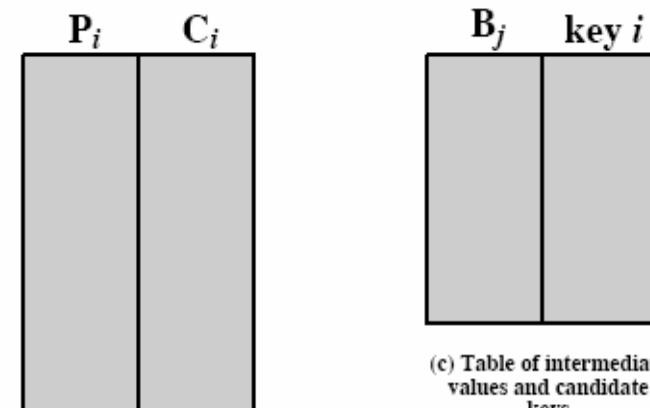


Schema di attacco (2/3)

- Per tutte le 2^{56} chiavi $K_i=j$ calcolare $B_j=D_j[a]$. Se si trova una corrispondenza nella tabella (c) si è trovata una coppia di chiavi candidate che produce una coppia nota (P, C).
- Verificare le coppie di chiavi candidate su altre coppie testo in chiaro/testo cifrato. Se necessario, ripetere la procedura con un nuovo valore di a .



(a) Two-key Triple Encryption with Candidate Pair of Keys

(b) Table of n known plaintext-ciphertext pairs, sorted on P

(c) Table of intermediate values and candidate keys

Schema di attacco (3/3)

- Date n coppie (P,C) , la probabilità di successo per un singolo valore selezionato di $A=a$ è $n/2^{64}$.
- Dalla teoria della probabilità è noto che il numero medio di tentativi per estrarre una pallina rossa da un'urna contenente n palline rosse e $(N-n)$ verdi è $(N+1)/(n+1)$.
- Pertanto il tempo si esecuzione dell'algoritmo è dell'ordine di

$$2^{56} \frac{2^{64}+1}{n+1} \approx 2^{56} \frac{2^{64}}{n} = 2^{120-\log_2 n}$$

Triple DES con tre chiavi

- La sicurezza di Triple DES può essere ulteriormente incrementata usando chiavi crittografiche distinte per ciascuna fase:

$$C = E_{K_3} [D_{K_2} [E_{K_1} [P]]]$$

- La compatibilità con DES si ha ponendo $K_3=K_2$ o $K_2=K_1$.
- Molte applicazioni per internet e per la gestione della posta elettronica hanno adottato Triple DES a tre chiavi (PGP, S/MIME, etc.).

I CAMPI FINITI

Campi finiti: Introduzione

- Ci occupiamo ora di “campi finiti”
- Rivestono un ruolo importante nella moderna crittografia
 - AES, curva ellittica, IDEA, chiave pubblica
- Avremo a che fare con operazioni su “numeri”
- Inizieremo con alcuni concetti di algebra astratta

Gruppo

- Un insieme di oggetti, elementi o “numeri”
- Un’operazione il cui risultato appartiene all’insieme (**chiusura**)
- proprietà:
 - associativa: $(a.b).c = a.(b.c)$
 - Esistenza dell’elemento identico e : $e.a = a.e = a$
 - Esistenza dell’ inverso a^{-1} : $a.a^{-1} = e$
- Se vale la proprietà commutativa $a.b = b.a$
 - Si parla allora di **gruppo abeliano**

Gruppo Ciclico

- Si definisca **esponenziazione** l'applicazione ripetuta dell'operazione
 - esempio: $a^{+3} = a \cdot a \cdot a$, $a^{-3} = a^{-1} \cdot a^{-1} \cdot a^{-1}$
- E si ponga: $e=a^0$
- Un gruppo è detto **ciclico** se ogni elemento si può ottenere come potenza di un elemento fisso, ovvero:
 - $b = a^k$ per qualche a and per ogni b nel gruppo
- a è detto **generatore** del gruppo

Anello

- Un insieme di “numeri” con due operazioni (addizione e moltiplicazione) che sono:
- Un gruppo abeliano per quanto riguarda l’addizione
- La moltiplicazione gode delle proprietà:
 - di chiusura
 - è associativa
 - è distributiva rispetto all’addizione: $a(b+c) = ab + ac$
- Se la moltiplicazione è commutativa si parla di **anello commutativo**
- Se esiste l’identità per l’operazione di moltiplicazione e non vi sono divisori dello zero, si parla allora di **dominio integrale**

Campo

- È un insieme di elementi su cui sono definite due operazioni (addizione e moltiplicazione)
- Gode di tutte le proprietà del dominio integrale, con in aggiunta la proprietà che ogni elemento, ad eccezione dello 0, ha un inverso moltiplicativo

Aritmetica Modulare

- L'operazione $a \bmod n$ da il resto intero della divisione di a per n
- Si dice che a è congruente a b modulo n e si indica come: $a \equiv b \pmod{n}$ se
 - a e b , divisi per n danno lo stesso resto
 - esempio $100 = 34 \pmod{11}$
- Tale resto è detto **residuo** di $a \bmod n$
- Tale resto è compreso tra 0 e $n-1$
 - $12 \bmod 7 \equiv -5 \bmod 7 \equiv 2 \bmod 7 \equiv 9 \bmod 7$

Esempio: Aritmetica Modulo 7

...

-21 -20 -19 -18 -17 -16 -15

-14 -13 -12 -11 -10 -9 -8

-7 -6 -5 -4 -3 -2 -1

0 1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31 32 33 34

...

Divisori

- Un numero b diverso da 0 **divide** a se il rapporto a/b è intero
- Si usa in tal caso la notazione $b | a$
- E si dice che b è un **divisore** di a
- Esempio: 1,2,3,4,6,8,12,24 dividono 24:
 $2|24$, $3|24$, $4|24$,

Operazioni in aritmetica modulare

- L'aritmetica modulare usa un numero finito di valori: le operazioni sono definite tutte modulo un certo intero
- Nota che vale la proprietà

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

Aritmetica Modulare

- Può essere realizzata con qualsiasi gruppo di interi: $Z_n = \{ 0, 1, \dots, n-1 \}$
 - L'insieme Z_n è un anello commutativo per l'addizione
 - Esiste anche l'identità nella moltiplicazione
 - Nota che
 - se $(a+b) \equiv (a+c) \pmod{n}$ allora $b \equiv c \pmod{n}$
 - ma $(ab) \equiv (ac) \pmod{n}$ allora $b \equiv c \pmod{n}$ solo se a è relativamente primo con n
- ES: $6x3=18 \equiv 2 \pmod{n}$, $6x7=42 \equiv 2 \pmod{n}$

Esempio: Aritmetica Modulo 8

+ \	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

Massimo Comun Divisore (Greatest Common Divisor)

- Il GCD (a,b) è il più grande intero che divide contemporaneamente a e b
 - esempio $\text{GCD}(60,24) = 12$
- Se due numeri non hanno fattori comuni, il loro GCD è 1 e i numeri sono detti **relativamente primi**
 - Esempio: $\text{GCD}(8,15) = 1$
 - Quindi, 8 e 15 sono relativamente primi

GCD: Algoritmo di Euclide

- È un modo efficiente per calcolare il GCD(a,b)
- Utilizza il risultato:
 - $\text{GCD}(a, b) = \text{GCD}(b, a \bmod b)$
- ***Algoritmo di Euclide***
 - A=a , B=b
 - while B>0
 - $R = A \bmod B$
 - $A = B, B = R$
 - return A

Esempio: GCD(1970,1066)

$$\begin{array}{lcl} 1970 = 1 \times 1066 + 904 & & \text{gcd}(1066, 904) \\ 1066 = 1 \times 904 + 162 & & \text{gcd}(904, 162) \\ 904 = 5 \times 162 + 94 & & \text{gcd}(162, 94) \\ 162 = 1 \times 94 + 68 & & \text{gcd}(94, 68) \\ 94 = 1 \times 68 + 26 & & \text{gcd}(68, 26) \\ 68 = 2 \times 26 + 16 & & \text{gcd}(26, 16) \\ 26 = 1 \times 16 + 10 & & \text{gcd}(16, 10) \\ 16 = 1 \times 10 + 6 & & \text{gcd}(10, 6) \\ 10 = 1 \times 6 + 4 & & \text{gcd}(6, 4) \\ 6 = 1 \times 4 + 2 & & \text{gcd}(4, 2) \\ 4 = 2 \times 2 + 0 & & \text{gcd}(2, 0) \end{array}$$



Campi di Galois

**Born: 25 Oct 1811 in Bourg La Reine (near Paris), France
Died: 31 May 1832 in Paris, France**



Evariste Galois: Breve biografia

Nato a Bourg-la-Reine nel 1811 da Nicholas Gabriel e Adelaide Marie Demante, due genitori interiggenti e con buone conoscenze nel campo della filosofia, della letteratura classica e della religione. La madre sarà la prima persona a curarsi della formazione del piccolo Évariste, fino all'età di 12 anni quando si iscrive al liceo *Louis-le-Grand*. I primi anni della sua carriera scolastica registrano buoni risultati in tutte le materie. Nel febbraio del 1827 Galois entra a far parte della sua prima classe di matematica di M. Vernier. Questi dirà di lui: "*E' la passione per la matematica a dominarlo, penso che sarebbe meglio per lui se i suoi genitori lo obbligassero a studiare qualsiasi altra cosa, sta sprecando il suo tempo qui e non fa altro che tormentare i suoi insegnanti e ricoprirsi di punizioni.*"

Le note scolastiche descriveranno spesso Galois come un ragazzo singolare, bizzarro, originale e chiuso. Nel 1828 Galois sostiene l'esame d'ammisione per l'*Ecole Polytechnique*, così torna al *Louis-le-Grand* però questa volta nella classe di Louis Richard. Lavora molto sulle proprie ricerche personali e molto poco sui propri compiti scolastici, tanto che Richard dice di lui: "*Questo studente lavora soltanto nei più alti regni della matematica*".

L'Aprile del 1829 vede la prima pubblicazione di un articolo di Galois sulle **frazioni continue** sui prestigiosi *Annales de mathématique*. Seguiranno a questa altre pubblicazioni sulla **risoluzione dell'equazioni algebriche**.

Ma continueranno ancora le difficoltà con le istituzioni scolastiche, infatti fallirà anche il suo secondo tentativo di entrare all'*Ecole Polytechnique* e dovrà 'rassegnarsi' ad entrare all'*Ecole Normale* di Parigi. Per entravi sostiene un esame al termine del quale il suo esaminatore di matematica riporterà la seguente nota:

"Questo ragazzo è talvolta oscuro nell'esprimere le sue idee, ma è intelligente e mostra un notevole spirito di ricerca"

Mentre il suo esaminatore di letteratura dirà:

"Questo è l'unico studente che mi ha risposto scarsamente, non conosce assolutamente nulla. Ho sentito dire che questo studente ha straordinarie capacità per la matematica. Ciò mi stupisce enormemente, poiché, dopo il suo esame, ritengo che possegga una scarsa intelligenza"

Galois spedisce a Cauchy i suoi lavori sulla teoria delle equazioni, ma viene a sapere di un articolo postumo di Abel che ricalca alcuni suoi risultati, così ritira i suoi lavori. Nel febbraio del 1830 invia a Cauchy un altro articolo *Sulle condizioni per cui un'equazione è risolvibile per radicali*, il suo lavoro viene girato a Fourier per considerarlo in vista del Gran Premio della Matematica, ma Fourier muore pochi mesi dopo e i lavori di Galois vanno misteriosamente perduti. Ma le disavventure di Galois non finiscono qua. Il giovane matematico verrà prima espulso dall'*Ecole Normale* e poi due volte arrestato per le sue fose e attivissima militanza politica repubblicana.

Galois si scontrò a duello con Perscheux d'Herbinville il 30 maggio 1832, le ragioni del duello non sono chiare, ma sembrano collegate in qualche modo ad una donna, Stephanie-Felice du Motel, di cui Galois si era invaghito qualche mese prima. Una sorta di leggenda racconta che Galois passò la notte prima del duello a scrivere tutto ciò che sapeva sulla **teoria dei gruppi**, è ragionevole pensare che ciò sia un po' esagerato. Comunque Galois fu gravemente ferito e lasciato sulla strada da d'Herbinville e dal suo secondo. Morirà in ospedale il giorno dopo, il 31 maggio 1832 quando non ha ancora compiuto i 21 anni.

Il materiale di Galois fu ricopiato e spedito dal fratello e da un amico a Gauss, Jacobi e altri. I matematici del tempo si accorgeranno del patrimonio rappresentato da quegli appunti solo una decina di anni dopo. Il contenuto di quei fogli passa oggi sotto il nome di **Teoria di Galois**: in meno di 21 anni di vita Evariste Galois portò alla luce concetti ancora fondamentali per l'algebra moderna.

Campi di Galois

- Si è visto che affinchè \mathbb{Z}_n possa essere un campo, una condizione sufficiente è che n sia un numero primo
- Galois introdusse campi finiti con un numero di elementi pari a p^n , ove p è un numero primo e n un intero positivo
- Tali campi sono detti **campi di Galois** e si indicano con $GF(p^n)$
- Useremo i campi:
 - $GF(p)$
 - $GF(2^n)$

Campo di Galois GF(p)

- GF(p) è l'insieme degli interi $\{0, 1, \dots, p-1\}$ con le usuali operazioni definite modulo p
- Tale insieme è un campo perchè ogni elemento ha il suo inverso moltiplicativo
- Quindi le operazioni matematiche sono “*well-behaved*” e si possono fare le 4 usuali operazioni +, -, x, / senza lasciare il campo

Esempio: GF(7)

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

Calcolo dell'inverso moltiplicativo

- Si usa l'algoritmo di Euclide esteso

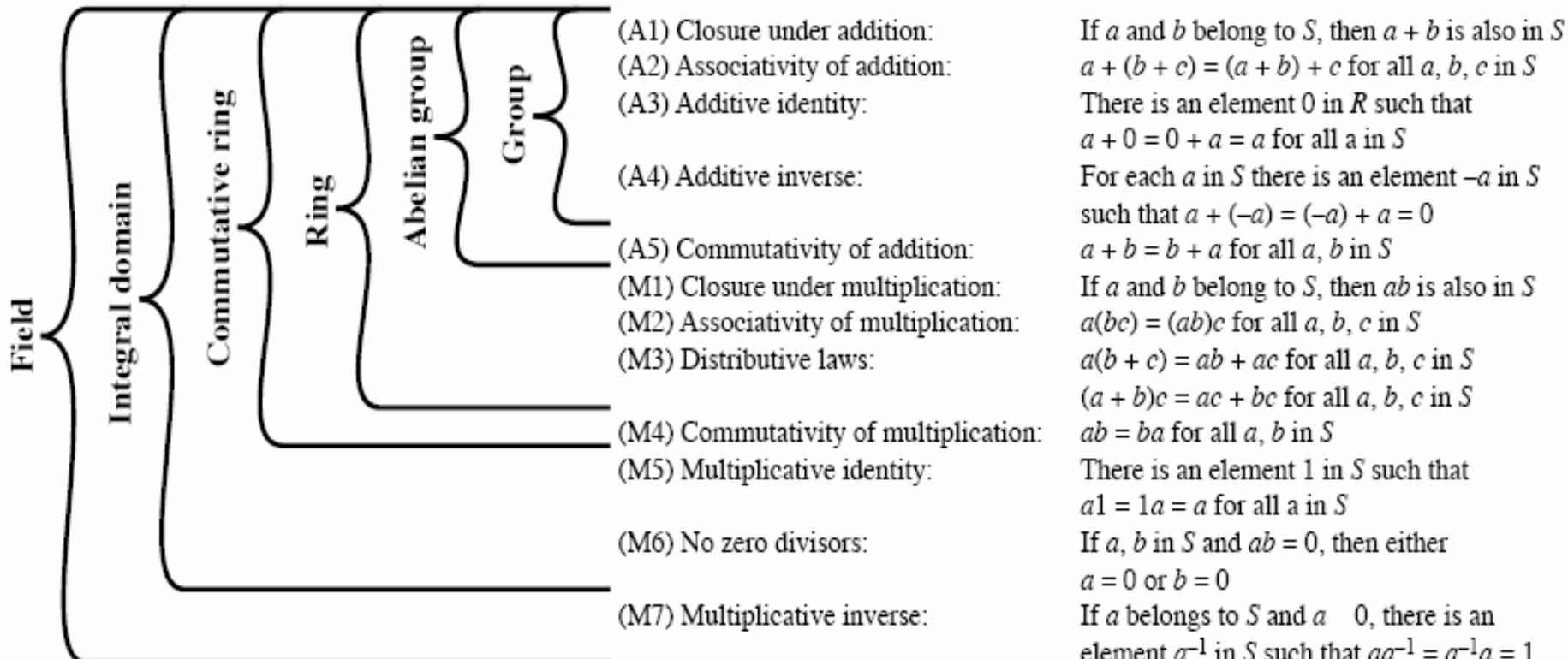
EXTENDED EUCLID(m, b)

1. $(A1, A2, A3) = (1, 0, m);$
 $(B1, B2, B3) = (0, 1, b)$
2. **if** $B3 = 0$
return $A3 = \text{gcd}(m, b);$ no inverse
3. **if** $B3 = 1$
return $B3 = \text{gcd}(m, b); B2 = b^{-1} \bmod m$
4. $Q = A3 \bmod B3$
5. $(T1, T2, T3) = (A1 - Q B1, A2 - Q B2, A3 - Q B3)$
6. $(A1, A2, A3) = (B1, B2, B3)$
7. $(B1, B2, B3) = (T1, T2, T3)$
8. **goto** 2

Esempio: calcolo inverso di 550 in GF(1759)

Q	A1	A2	A3	B1	B2	B3
-	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	355	1

Riepilogo



Riepilogo

- **Campo:** è un insieme su cui si possono eseguire le operazioni di + , - , x , / senza uscire fuori dall'insieme.
 - Campi di ordine infinito sono i numeri reali, i numeri razionali e i numeri complessi.
 - $Z_p = \{ 0, 1, \dots, p-1 \}$, con p numero primo, è un campo se le operazioni vengono eseguite modulo p .
- In crittografia siamo interessati a lavorare su campi di ordine finito con un numero di elementi che sia una potenza di due.

Riepilogo

- È possibile costruire campi finiti di ordine P^n , con P numero primo e n intero (Galois Field).
 - Z_{2^n} non è un campo per $n>1$.
- I campi $GF(P^n)$ con $n>1$ hanno una struttura differente da Z_p .
 - Le operazioni su questi campi seguono le regole dell'aritmetica polinomiale modulare su $GF(P)$
- Per studiare questi campi è necessario introdurre alcuni concetti sull'aritmetica dei polinomi.

Aritmetica Polinomiale

- Fa uso di polinomi

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

- Vi sono varie alternative
 - Aritmetica ordinaria
 - Aritmetica polinomiale con coefficienti mod p
 - Aritmetica polinomiale con coefficienti mod p e polinomi mod M(x)

Aritmetica Polinomiale Ordinaria

- Si utilizzano le regole convenzionali
- Ad esempio

– sia $f(x) = x^3 + x^2 + 2$ e $g(x) = x^2 - x + 1$

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$

$$f(x) - g(x) = x^3 + x + 1$$

$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

Aritmetica polinomiale con coefficienti modulo p

- I coefficienti sono scalati modulo p
- L'insieme dei polinomi è in tal caso un anello (anello polinomiale)
- p può essere qualsiasi numero primo
- Ci interessa tuttavia il caso p=2
 - Ovvero, tutti i coefficienti sono 0 o 1
 - La somma equivale allo XOR, la moltiplicazione all'AND
 - Es.: sia $f(x) = x^3 + x^2$ e $g(x) = x^2 + x + 1$
 $f(x) + g(x) = x^3 + x + 1$
 $f(x) \times g(x) = x^5 + x^2$

Aritmetica polinomiale con coefficienti modulo p

- Anche se l'insieme dei coefficienti è un campo, la divisione fra polinomi non sempre è esatta. In generale, la divisione produce un quoziente ed un resto:

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)} \Leftrightarrow f(x) = q(x)g(x) + r(x)$$

- Se il grado di $f(x)$ è n ed il grado di $g(x)$ è m ($m \leq n$), allora il grado del quoziente $q(x)$ è $n-m$ ed il grado del resto è al più $m-1$.

Aritmetica polinomiale con coefficienti modulo p

- L'insieme di tutti polinomi i cui coefficienti sono elementi di un campo forma un **dominio integrale** (nota analogia con l'insieme dei **numeri interi**).
- In analogia con l'aritmetica degli interi, si definisce $r(x)=f(x) \text{ mod } g(x)$.
- Se $r(x)=0$, allora $g(x)$ divide $f(x)$.
- se $g(x)$ ha come soli divisori se stesso e 1, allora si dice essere un polinomio **irriducibile** (o primo).

Esempio

- Il polinomio $f(x)=x^4+1$ su GF(2) è riducibile:
 - $x^4+1=(x+1)(x^3+x^2+x+1)$
- Il polinomio $f(x)=x^3+x+1$ su GF(2) è irriducibile:
 - Non è divisibile per x .
 - Non è divisibile per $x+1$.
 - $f(x)$ non ha fattori di grado 1. Se fosse riducibile dovrrebbe avere un fattore di grado 1 ed uno di grado 2.

Massimo Comune Divisore di polinomi

- $c(x) = \text{GCD}(a(x), b(x))$ se $c(x)$ è il polinomio di grado massimo che divide sia $a(x)$ che $b(x)$
- Si utilizza l'algoritmo di Euclide per il calcolo
 - **EUCLID**[$a(x)$, $b(x)$]
 1. $A(x) = a(x); B(x) = b(x)$
 2. **if** $B(x) = 0$ **return** $A(x) = \text{gcd}[a(x), b(x)]$
 3. $R(x) = A(x) \bmod B(x)$
 4. $A(x) \leftarrow B(x)$
 5. $B(x) \leftarrow R(x)$
 6. **goto** 2

Aritmetica polinomiale modulare

- Si consideri l'insieme S di tutti i polinomi di grado $n-1$ con coefficienti in \mathbb{Z}_p
$$f(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\dots+a_1x+a_0$$
- Si considerino le seguenti regole **dell'aritmetica modulare**:
 1. Si seguono le regole dell'aritmetica ordinaria fra polinomi con due raffinamenti successivi.
 2. L'aritmetica sui coefficienti viene eseguita modulo p .
 3. Se la moltiplicazione produce un polinomio di grado maggiore di $n-1$, viene ridotto modulo un polinomio irriducibile $m(x)$ di grado n . In pratica, si divide per $m(x)$ e si tiene il resto.

Aritmetica polinomiale modulare

L'insieme S con le regole
dell'aritmetica modulare forma
un campo, $\text{GF}(P^n)$

IL Campo GF(2^n)

- Sono polinomi con coefficienti modulo 2
- Il loro grado è minore di n
- Quindi devono essere ridotti in modulo tramite un polinomio irriducibile di grado n
- Formano un campo finito
- Quindi esiste sempre l'elemento inverso
 - L'algoritmo esteso di Euclide permette il calcolo dell'inverso
- Ogni polinomio è in corrispondenza con una stringa di n bit!!! (ecco perchè li studiamo)

Esempio GF(2³)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

		000	001	010	011	100	101	110	111
	+	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+1	x^2	$x+1$	x	1	0	0

(a) Addition

		000	001	010	011	100	101	110	111
	*	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) Multiplication

Polinomio irriducibile $x^3 + x + 1$

Esempio

AES usa l'aritmetica di $GF(2^8)$, con polinomio irriducibile

$$m(x)=x^8 + x^4 + x^3 + x^8 + 1$$

NOTA: Per costruire $GF(2^8)$, infatti, c'è bisogno di scegliere un polinomio irriducibile di grado 8

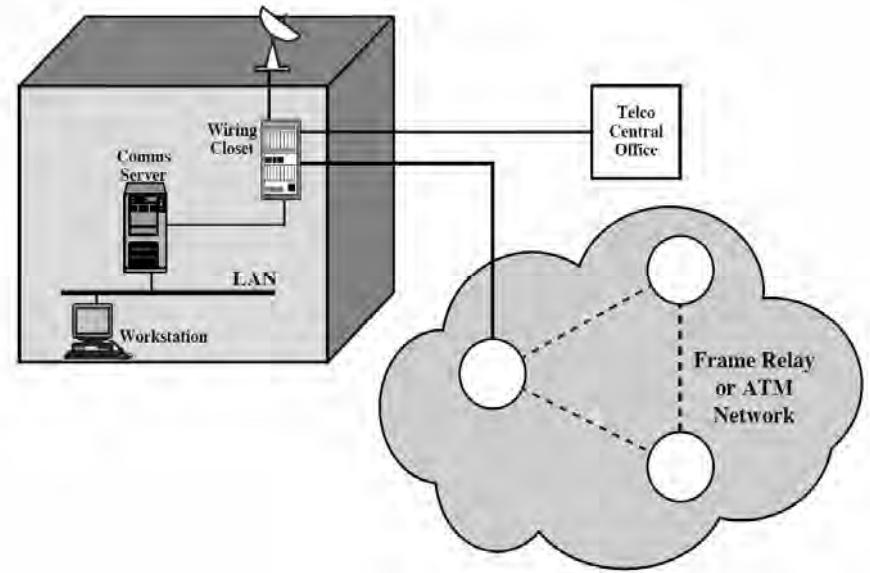
Segretezza e crittografia simmetrica

Sommario

- Potenziali punti di attacco
- Posizionamento della logica crittografica
 - Crittografia del collegamento
 - Crittografia punto-a-punto
- Mascheramento del traffico
- Distribuzione della chiave
- Cenni sulla generazione di numeri casuali

Scenario tipico

- Workstation e Server connessi fra loro mediante reti LAN;
- Reti LAN di uno stesso edificio connesse fra loro mediante switch e router;
- Reti LAN geograficamente separate interconnesse mediante reti pubbliche a commutazione di pacchetto.



Possibili attacchi alla segretezza

- Analisi del traffico all'interno della rete locale (dall'interno o dall'esterno).
- Attacco all'armadio di cablaggio e/o ad uno qualsiasi dei collegamento fisici della rete.
- Analisi e/o alterazione dei pacchetti in transito nei nodi di commutazione delle reti di interconnessione pubbliche.

Tipi di attacco

- Attacco attivo
 - L'intruso deve assumere il controllo fisico di una porzione di collegamento (in genere è necessario far uso di apparecchiature invasive).
 - L'intruso è in grado sia di ascoltare che di modificare le trasmissioni.
- Attacco passivo
 - L'intruso non ha il controllo fisico del canale, ma è in grado di ascoltare le trasmissioni (utilizzando mezzi non invasivi).

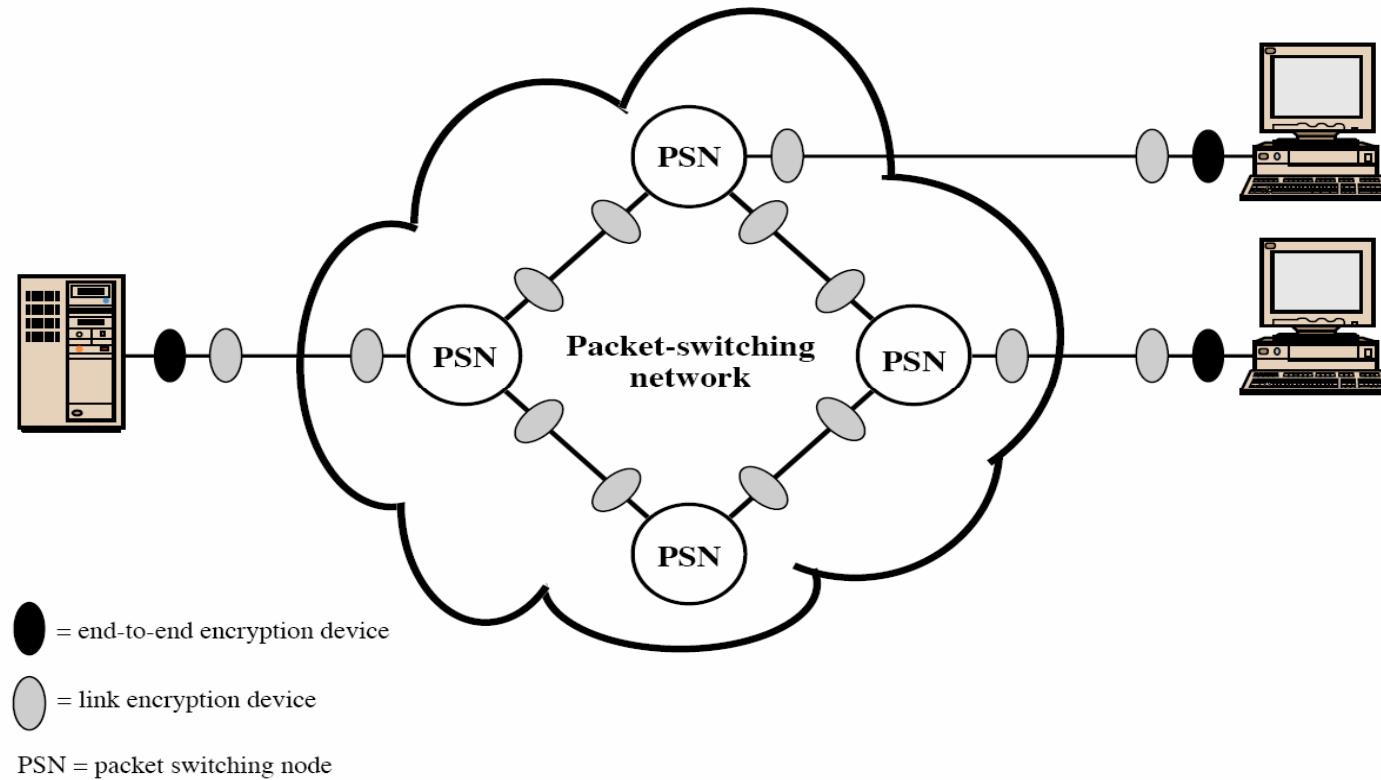
Crittografia del collegamento

- Ciascun collegamento fisico vulnerabile viene dotato di un dispositivo di crittografia.
- Vantaggi:
 - Tutto il traffico su tutti i collegamenti è sicuro
 - Ogni coppia di nodi ha una chiave univoca usata per tutti i messaggi in transito
 - Facilità nella distribuzione delle chiavi
- Svantaggi:
 - Il messaggio è decrittografato (e dunque vulnerabile) negli switch
 - L'utente non controlla la sicurezza dei nodi della rete pubblica
 - Tutti i collegamenti devono essere crittografati
 - Assenza di autenticazione

Crittografia punto-a-punto

- Il processo di crittografia viene eseguito solo dai due sistemi terminali, ed è completamente trasparente ai nodi interni della rete di interconnessione.
- Vantaggi:
 - Dati protetti contro gli attacchi ai collegamenti o agli switch
 - Garantisce l'autenticazione
- Svantaggi:
 - La distribuzione delle chiavi è più complessa
 - L'indirizzo del destinatario non può essere crittografato

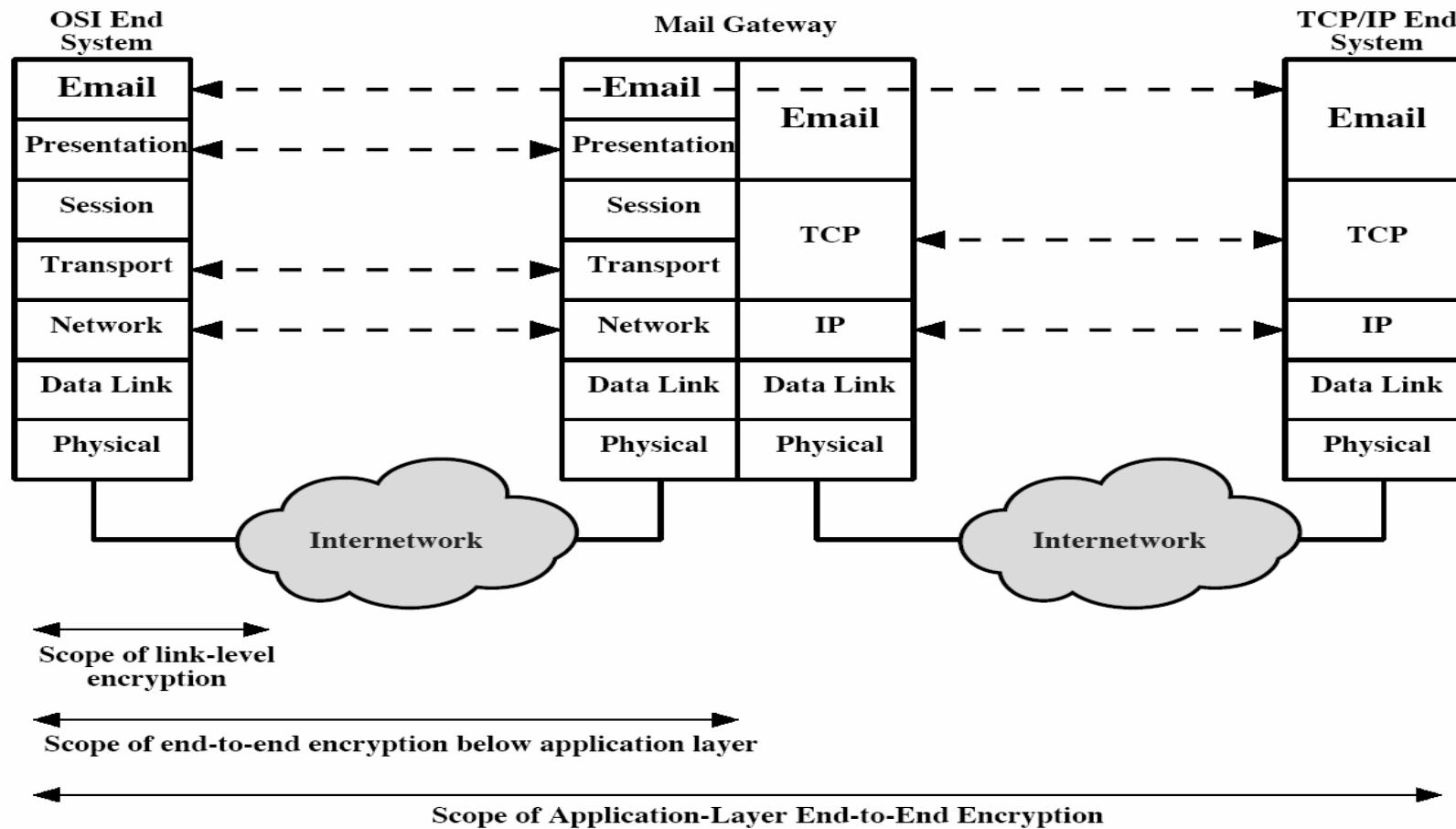
Crittografia del collegamento e punto-a-punto



Posizionamento logico della crittografia

- Crittografia del collegamento: livello fisico (1) o di collegamento (2) del modello OSI (Open System Interconnection).
- Crittografia punto-a-punto: livelli 3-7 del modello OSI.
- Spostandoci verso l'alto nella gerarchia dei livelli di comunicazione vengono crittografate meno informazioni, ma si ottiene una maggiore sicurezza sui dati. Inoltre, aumenta il numero di chiavi crittografiche da utilizzare.

Posizionamento logico della crittografia



Posizionamento logico della crittografia



(a) Application-Level Encryption (on links and at routers and gateways)



On links and at routers



In gateways

(b) TCP-Level Encryption



On links



In routers and gateways

(c) Link-Level Encryption

Shading indicates encryption.

TCP-H = TCP header
IP-H = IP header
Net-H = Network-level header (e.g., X.25 packet header, LLC header)
Link-H = Data link control protocol header
Link-T = Data link control protocol trailer

Utilizzo delle chiavi

- **Crittografia del collegamento:** una singola chiave è utilizzata per crittografare tutti i dati in transito nel collegamento.
- **Crittografia punto-a-punto a livello di rete:** tutti i processi utenti e le applicazioni di un dato terminale T1 condividono la stessa chiave per comunicare con un altro terminale T2.
- **Crittografia punto-a-punto a livello dell'applicazione:** ogni processo utente o applicazione di un dato terminale T1 usa una propria chiave per comunicare con il corrispondente processo utente o applicazione del terminale T2

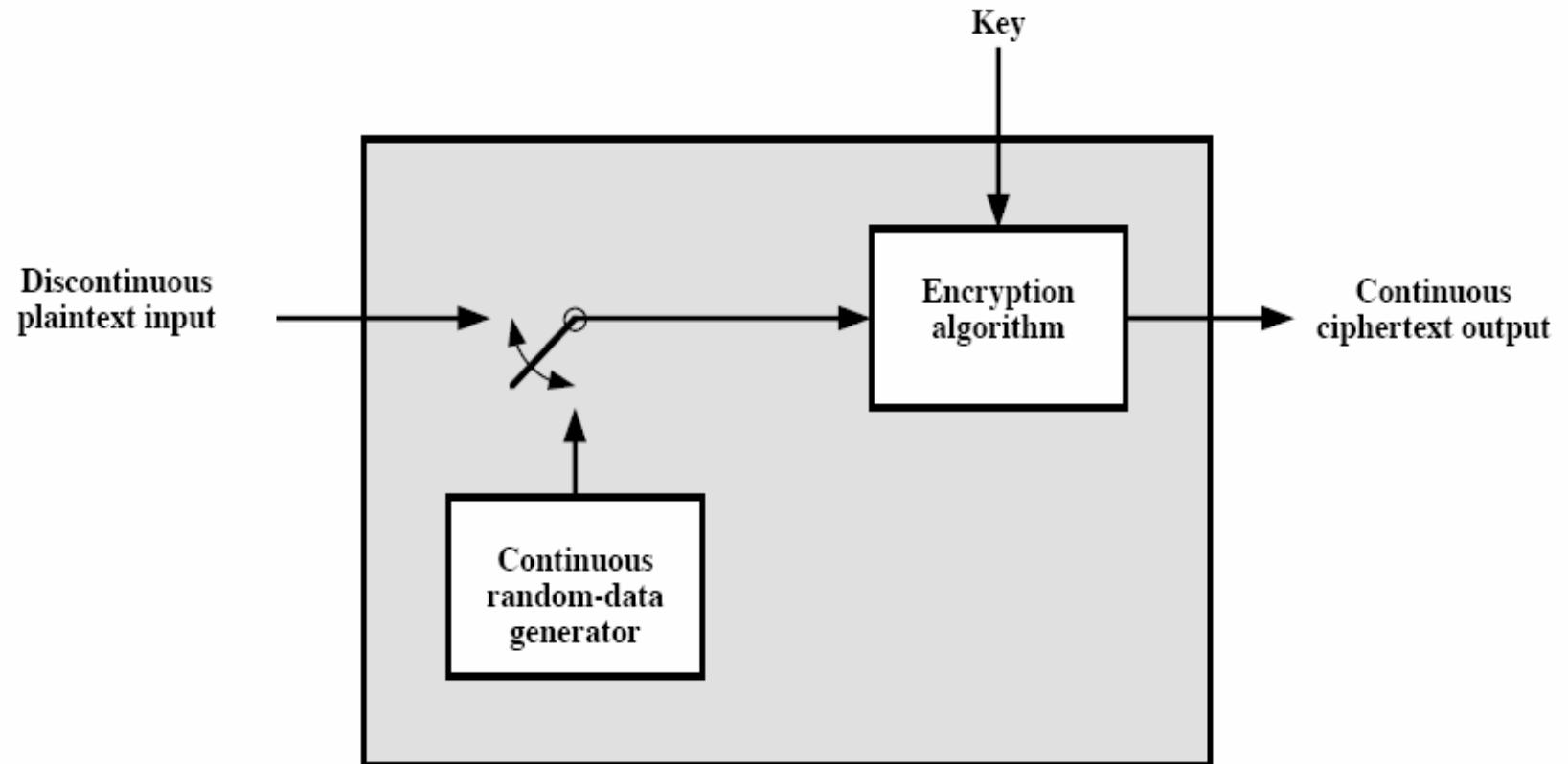
Analisi del traffico

- L'analisi del traffico può fornire informazioni su:
 - Identità dei partner
 - Frequenza di comunicazione dei partner
 - Lunghezza e/o quantità dei messaggi scambiati
 - Schema di trasmissione dei messaggi
- Schemi di traffico segreti possono essere impiegati per creare canali nascosti (ad esempio, pacchetto lungo = 0, pacchetto corto = 1).
- L'analisi del traffico è usata in ambito militare e commerciale.

Segretezza del traffico

- La **crittografia punto-a-punto** è estremamente vulnerabile all'analisi del traffico (essendo le intestazione dei pacchetti in chiaro). Contromisure:
 - inserimento di pacchetti nulli
 - pacchetti in uscita di uguale dimensione (maschera il volume di dati scambiato)
 - crittografia del collegamento (maschera gli indirizzi finali)
- La **crittografia del collegamento** può essere resa più sicura utilizzando la tecnica **Traffic Padding**: in assenza di testo in chiaro viene comunque prodotto un output cifrato in maniera casuale.

Traffic Padding



Distribuzione della chiave

- Dati due terminali A e B, si possono avere varie alternative per la distribuzione delle chiavi.
 1. A sceglie una chiave e la consegna fisicamente a B;
 2. Una parte terza C sceglie una chiave e la consegna fisicamente ad A e B;
 3. Se A e B hanno usato recentemente una chiave, una delle parti trasmette all'altra parte la nuova chiave usando la vecchia chiave;
 4. Se A e B hanno una connessione crittografata sicura con C, quest'ultima può consegnare una chiave sui collegamenti crittografati ad A e B;

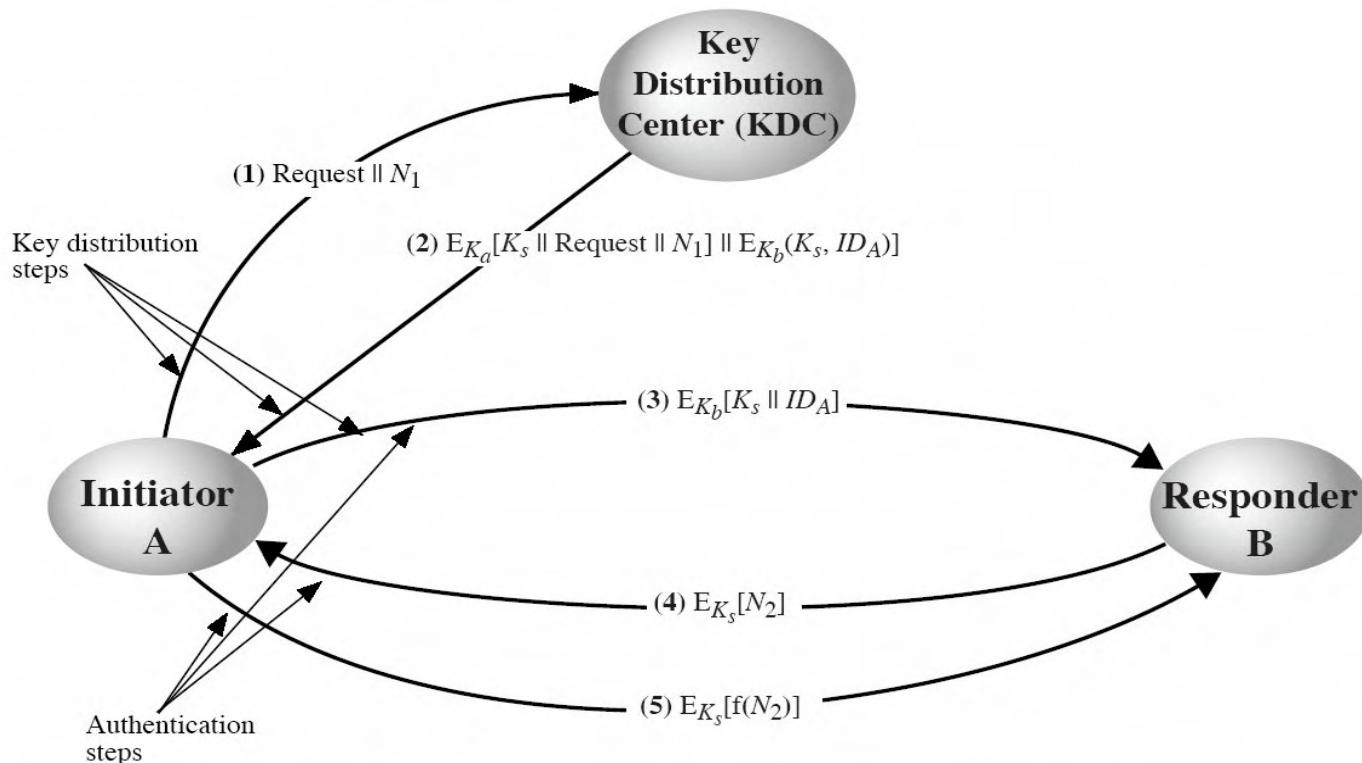
Distribuzione della chiave

- Le opzioni 1 e 2 richiedono la consegna manuale della chiave: sono ragionevoli solo per la crittografia di un collegamento.
- L'opzione 3 è poco sicura: se una chiave viene violata, anche le chiavi successive saranno automaticamente violate.
- Per la crittografia punto-a-punto sono generalmente adottate delle varianti dell'opzione 4.

Centro di Distribuzione delle Chiavi (KDC)

- Per la crittografia punto-a-punto si assume che ci sia un centro responsabile della distribuzione delle chiavi alle coppie di utenti in base alle richieste.
- Sono utilizzati due livelli gerarchici di chiavi
 - **Chiavi master**: condivise fra ciascun utente e il KDC
 - **Chiavi di sessione**: utilizzate per la comunicazione fra due utenti
- Le chiave di sessione vengono trasmesse alle coppie di utenti in forma crittografata usando le chiavi master.
- Il problema della distribuzione delle chiavi è semplificato: se ci sono N terminali andranno consegnate fisicamente solamente N chiavi master.

Esempio di distribuzione della chiave di sessione



- $K_A \rightarrow$ chiave master di A (nota solo ad A ed al KDC)
- $K_B \rightarrow$ chiave master di B (nota solo ad B ed al KDC)

Controllo gerarchico della chiave

- In reti di grandi dimensioni si possono definire centri gerarchici di distribuzione delle chiavi.
 - Ogni centro è responsabile di un piccolo dominio
 - Per comunicazioni fra terminali dello stesso dominio, le richieste vengono gestite dal KDC locale
 - Per comunicazione fra terminali in domini differenti, i KDC locali comunicano con un KDC globale per coordinare la scelta della chiave
- Vantaggi: maggiore robustezza ai guasti e semplicità nella distribuzione delle chiavi master.

Durata di una chiave di sessione

- Due esigenze contrastanti:
 - Maggiore è la frequenza con cui si cambiano le chiavi di sessione, maggiore sarà la sicurezza delle chiavi.
 - La distribuzione delle chiavi introduce un overhead iniziale ad ogni comunicazione, introducendo ritardi di trasmissioni. Inoltre, genera traffico nella rete.
- Regola pratica: vanno stabiliti **tempi massimi** di durata e/o **numero massimo di transazioni** effettuabili con una singola chiave di sessione.

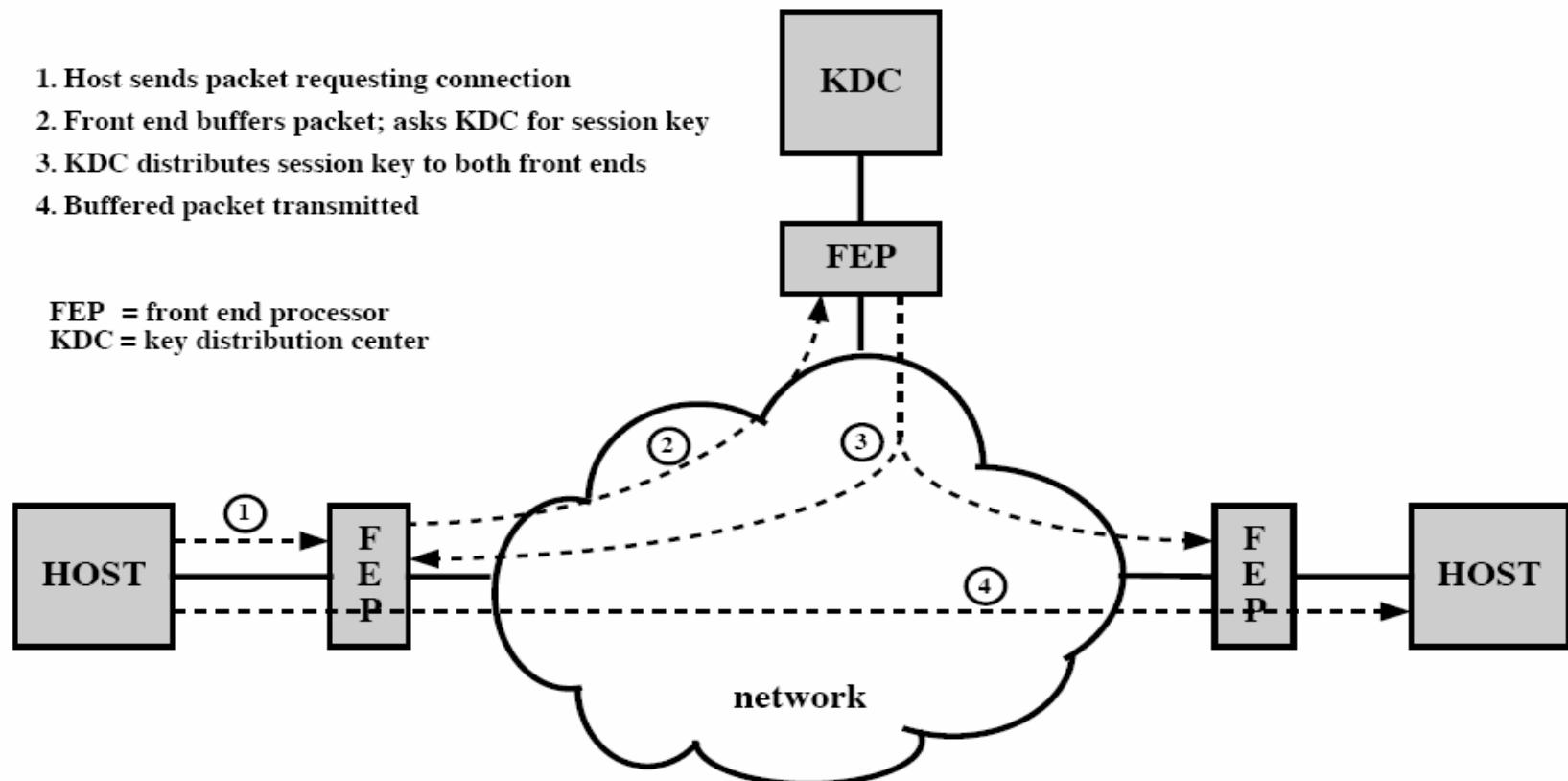
Schema a controllo trasparente della chiave

- Per crittografie punto-a-punto a livello di rete orientate alla connessione, le operazioni di richiesta della chiave di sessione e di crittografia del messaggio possono essere delegate ad un processore esterno denominato FEP (Front-End-Processor).
- Con quest'approccio l'operazione di crittografia diviene trasparente ai singoli terminali:
 - Per il terminale e la rete esterna, il processore FEP appare come un nodo di commutazione

Schema a controllo trasparente della chiave

1. Host sends packet requesting connection
2. Front end buffers packet; asks KDC for session key
3. KDC distributes session key to both front ends
4. Buffered packet transmitted

FEP = front end processor
KDC = key distribution center

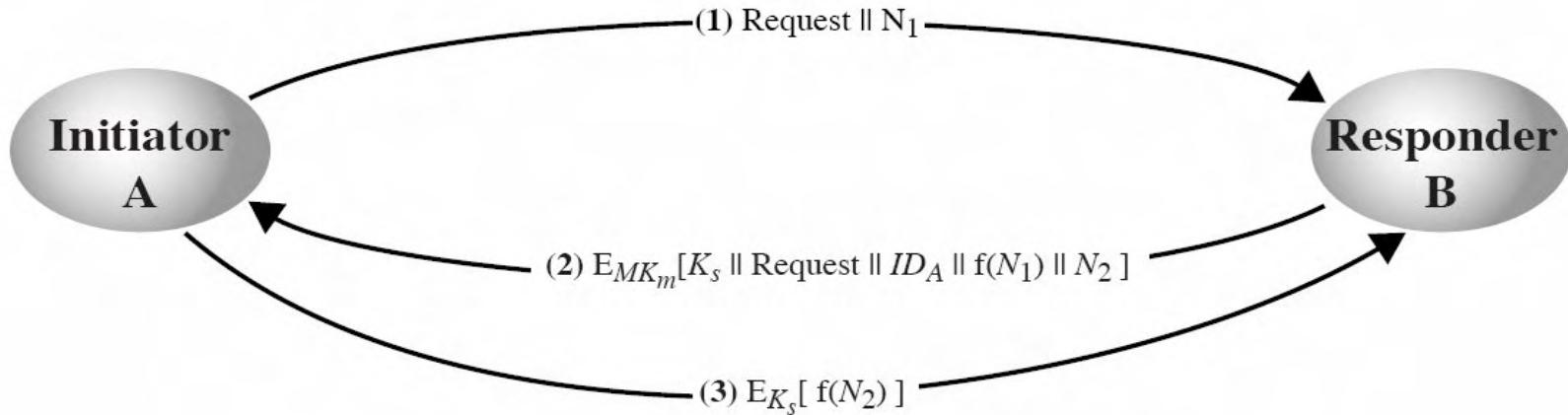


Controllo della chiave decentralizzato

- In uno schema centralizzato, il KDC deve essere fidato e protetto contro ogni tipo di attacco.
 - I rischi derivanti da eventuali attacchi al KDC potrebbero essere eliminati se il processo di distribuzione delle chiavi di sessione fosse decentralizzato.
- Una decentralizzazione completa in reti di grandi dimensioni non è pensabile. Tuttavia, è una valida alternativa in contesti locali.

Controllo della chiave decentralizzato

- Si assume che ci siano n terminali in grado di comunicare in modo univoco fra loro.
 - Sono richieste $n(n-1)/2$ chiavi master (la distribuzione può avvenire di persona dato che queste chiavi non verranno più cambiate).
- Le chiavi di sessione vengono ottenute come segue:



Controllo dell'utilizzo delle chiavi

- Il concetto di gerarchia delle chiavi e l'uso di tecniche automatiche di distribuzione delle chiavi riduce notevolmente il numero di chiavi da distribuire manualmente.
- Per un corretto uso delle chiavi sessione distribuite dai KDC, può essere utile specificare la loro funzione:
 - Chiavi di sessione / Chiavi master
 - Chiavi di crittografia per dati
 - Chiavi di crittografia per codici PIN
 - Chiavi per crittografia di file
 - ...

Esempio di controllo dell'utilizzo delle chiavi

- DES accetta in ingresso una chiave a 64 bit. Tuttavia, solo 56 bit vengono utilizzati per produrre le sottochiavi. I restanti 8 bit hanno il seguente significato:
 - Un bit indica se la chiave è di sessione o master
 - Un bit indica se la chiave può essere utilizzata per la crittografia
 - Un bit indica se la chiave può essere utilizzata per la decrittografia
 - I bit rimanenti sono riservati per utilizzi futuri

Generazione di numeri casuali

- I numeri casuali vengono utilizzati in numerosi algoritmi crittografici:
 - Schemi di autenticazione
 - Generazione chiave di sessione
 - Generazione chiavi per l'algoritmo di crittografia RSA a chiave pubblica (che verrà studiato in seguito)
- Idealmente gli elementi di una sequenza casuale sono statisticamente indipendenti fra loro (ovvero, imprevedibili) ed uniformemente distribuiti.

Generazione di numeri casuali

- Trovare sorgenti di numeri effettivamente casuali è difficile.
 - I generatori fisici di rumore sono buone sorgenti, ma sono poco pratici da usare.
- Un'alternativa è usare algoritmi numerici deterministici per la produzione di sequenze di numeri pseudo-casuali.
 - Se l'algoritmo è di buona qualità, le sequenze prodotte passeranno numerosi test ragionevoli di casualità

Metodo della congruenza lineare

$$X_{n+1} = (aX_n + c) \bmod m$$

m il modulo $m > 0$

a il moltiplicatore $0 < a < m$

c l'incremento $0 \leq c \leq m$

X_0 il valore iniziale o seme $0 \leq X_0 \leq m$

Metodo della congruenza lineare

- La scelta dei valori di a , c , e m è fondamentale per ottenere buone sequenza di numeri pseudo-casuali.
- Si vorrebbe che m fosse molto esteso in modo da produrre una lunga sequenza di numeri casuali distinti
 - m è scelto pari al massimo intero non-negativo rappresentabile
- Si dimostra che una buona scelta è $a=16807$, $c=0$ e $m=2^{31}-1$
 - La sequenza che si ottiene ha un periodo pari ad m

Metodo della congruenza lineare

- Una volta fissati il seme ed i parametri a e c , la sequenza risultante di numeri segue in maniera deterministica.
- Nell'algoritmo non vi è nulla di casuale, se non la scelta dei parametri iniziali.
- La sequenza di numeri così generata può essere ricostruita da un estraneo che è a conoscenza dell'algoritmo e dei parametri usati.
 - Per evitare tale problema la sequenza viene periodicamente riavviata usando come seme il valore dell'orologio interno del calcolatore.

Algoritmo AES (Advanced Encryption Standard)

Origini di AES

- 3DES non rappresenta un buon candidato nel lungo periodo:
 - Esecuzione software lenta (DES era stato progettato agli inizi degli anni '70 per un'implementazione hardware)
 - Usa blocchi di 64 bit di dati (per motivi di efficienza e sicurezza sarebbe preferibile utilizzare blocchi di lunghezza maggiore)
- Nel 1997 il NIST emette una richiesta di proposte per un nuovo standard chiamato (AES).
- Il nuovo standard dovrà progressivamente sostituire 3DES.

Origini di AES

- Requisiti minimi richiesti:
 - Doti di sicurezza uguali o maggiori a 3DES
 - Elevata efficienza computazionale
 - Blocchi dati di 128 bit / Lunghezza chiavi: 128, 192, 256 bit
 - Specifiche di progetto di pubblico dominio
 - Vita media 20-30 anni
- Inizialmente vengono accettate 15 delle 21 proposte.
- Successivamente i candidati sono ristretti a 5.
- Standard finale (FIPS PUB 197) pubblicato nel 2001. L'algoritmo selezionato è 'RIJNDAEL', sviluppato dai crittografi belgi Dr. Joan Daemen e Dr. Vincent Rijmen.

Criteri iniziali di valutazione di AES

- **Sicurezza:** Scartati gli attacchi a forza bruta, viene valutato l'impegno necessario per un analisi crittografica dell'algoritmo.
- **Costo:** algoritmo disponibile a tutti, ed utilizzabile in un ampia gamma di applicazioni; criteri di valutazione sono anche l'efficienza computazionale e i requisiti di memoria.
- **Caratteristiche dell'algoritmo e dell'implementazione:** flessibilità, possibilità di usare chiavi e blocchi di dati di varie dimensioni, possibilità di impiego in varie implementazioni hardware e software, semplicità di analisi

AES: caratteristiche

- **Sicurezza Generale:** basato su un'analisi pubblica della sicurezza condotta dalla comunità dei crittografi (durata 3 anni).
- **Implementazioni software:** velocità di esecuzione, prestazioni su piattaforme differenti, velocità in funzione della lunghezza della chiave.
- **Ambienti con spazio limitato:** occupazione di memoria
- **Implementazione hardware:** costi, velocità di esecuzione e dimensioni dei chip.
- **Attacchi alle implementazioni:** resistenza ad attacchi che misurano il tempo di esecuzione e/o la potenza consumata.

AES: Caratteristiche

- **Crittografia e Decrittografia:** differenze fra gli algoritmi di crittografia e decrittografia.
- **Agilità della chiave:** tempo richiesto per impostare una nuova chiave, calcolo delle sottochiavi
- **Flessibilità:** possibilità di variare e/o ottimizzare i parametri dell'algoritmo in base al tipo di applicazione richiesta.
- **Parallelismo:** capacità di sfruttare le funzionalità di architetture multi-processore.

Cifratura AES

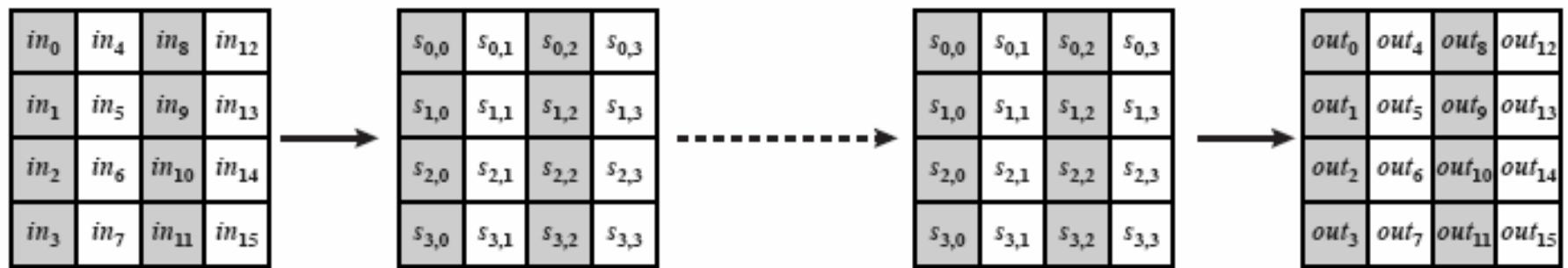
- Criteri progettuali dell'algoritmo RIJNDAEL:
 - Resistenza contro attacchi noti
 - Velocità e compattezza del codice
 - Semplicità
- Blocco dati: 128 bit / Lunghezza della chiave: qualunque multiplo di 32 bit.
 - In pratica si usano chiavi di **128**, 192 e 256 bit
- I parametri dell'algoritmo dipendono dalla lunghezza della chiave (vedi lucido seguente).
- Nel seguito si considera una chiave a 128 bit.

Parametri Algoritmo

Tabella 5.3 I parametri di AES.

Dimensioni della chiave (word/byte/bit)	4/16/128	6/24/192	8/32/256
Dimensioni del blocco di testo in chiaro (word/byte/bit)	4/16/128	4/16/128	4/16/128
Numero di fasi	10	12	14
Dimensioni della chiave di fase (word/byte/bit)	4/16/128	4/16/128	4/16/128
Dimensioni espansse della chiave (word/byte)	44/176	52/208	60/240

Dati Input Algoritmo AES

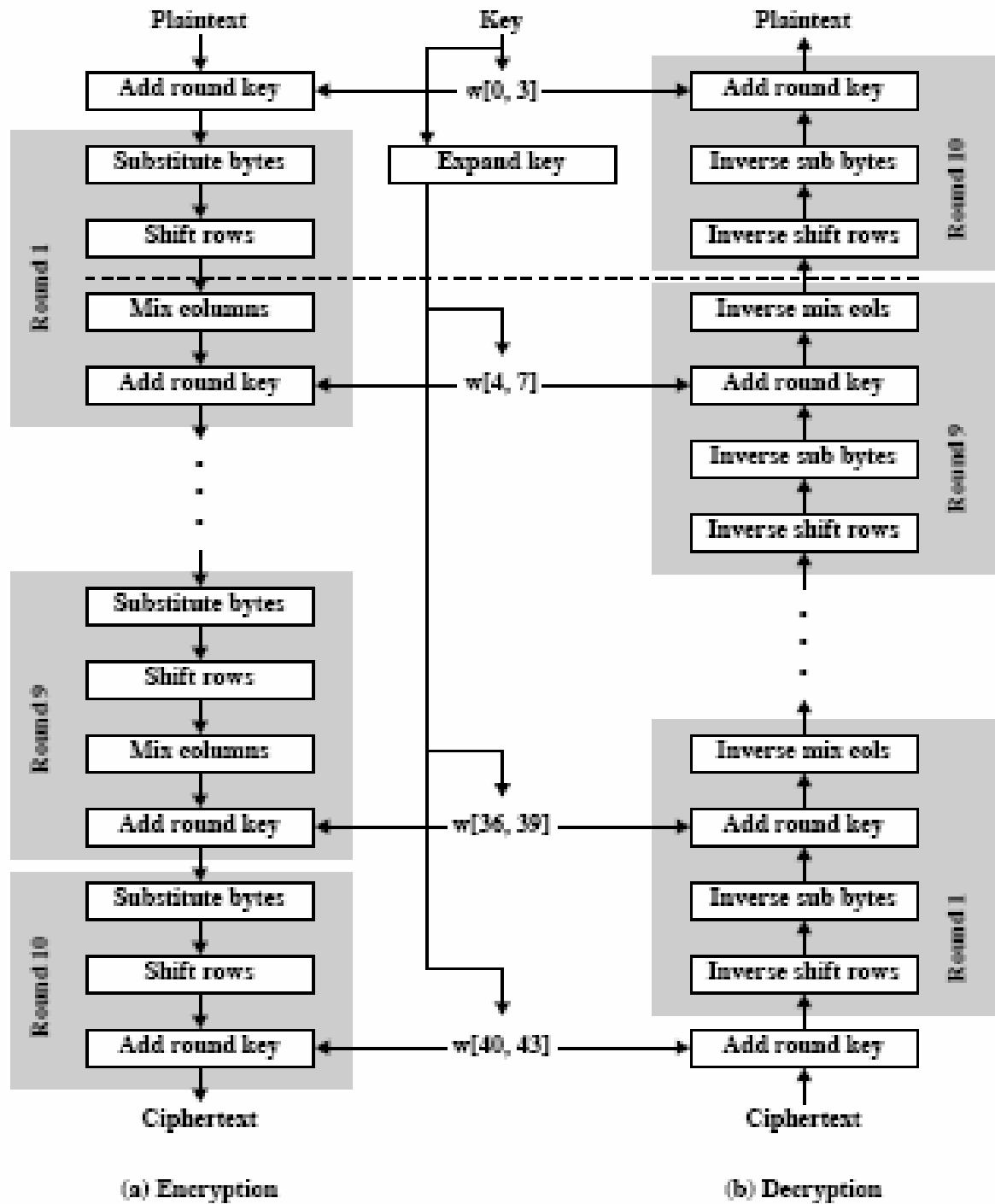


(a) Input, state array, and output



(b) Key and expanded key

Schema Algoritmo AES



Commenti su AES

- Non si tratta di una struttura di Feistel.
- La chiave viene espansa in 44 word da 32 bit. Ciascuna fase usa 4 word distinte.
- Operazioni effettuate
 - **Substitute Bytes:** blocchi di 8 bit sono sostituiti mediante una S-box.
 - **Shift Rows:** semplice permutazione.
 - **Mix Columns:** sostituzione che usa l'aritmetica su GF(2^8), con il polinomio irriducibile $m(x)=x^8+x^4+x^3+x+1$.
 - **Add Round Key:** XOR bit-a-bit con la sottochiave.

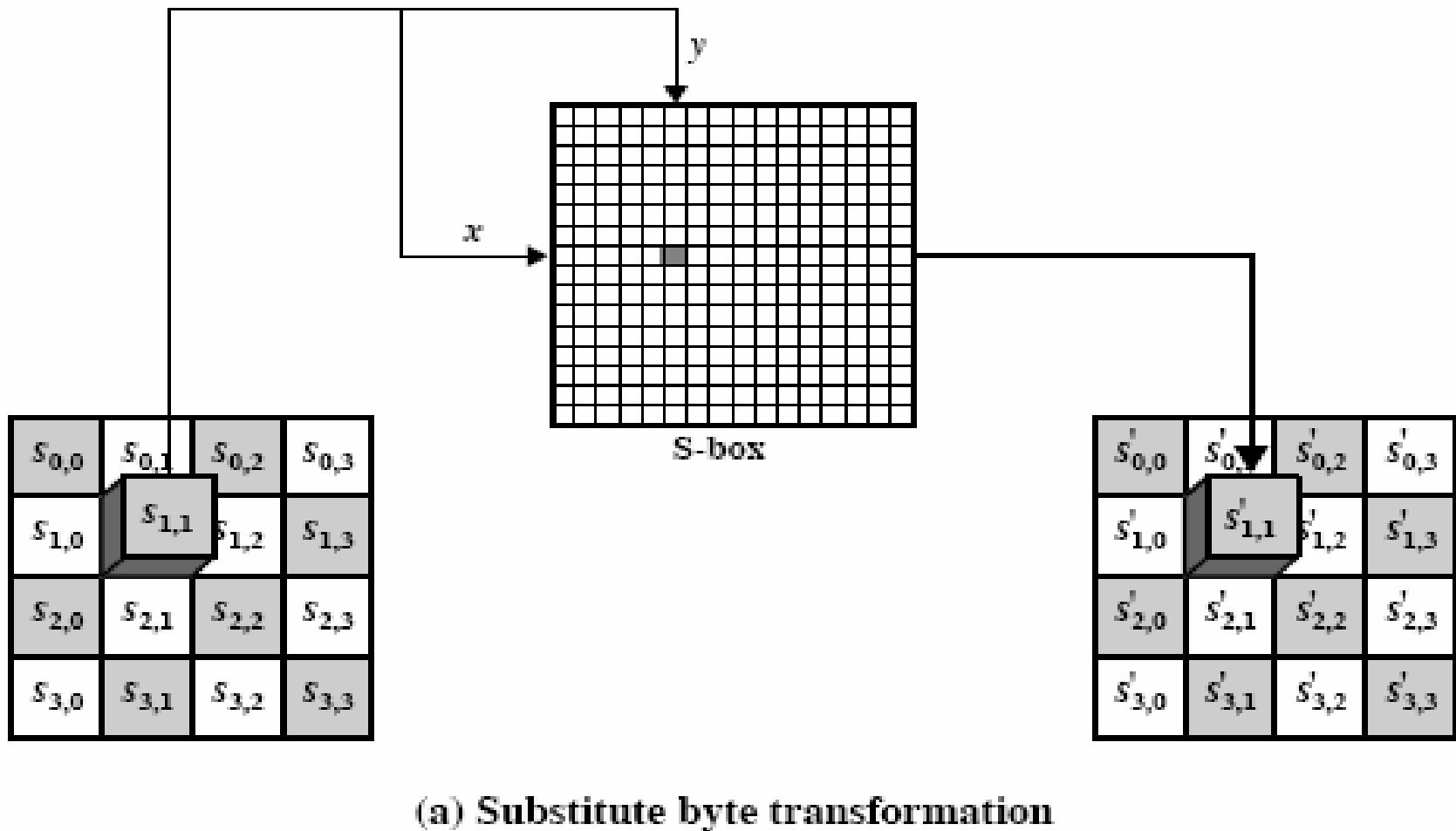
Commenti su AES

- Tutte le trasformazioni sono invertibili.
- L'algoritmo di decrittografia usa le sottochiavi in ordine inverso.
- L'algoritmo di decrittografia non è identico a quello di crittografia.
- L'ultima fase della crittografia e della decrittografia contiene solo tre trasformazioni.

Trasformazione Substitute Bytes (diretta)

- Viene definita una matrice 16x16 di byte detta S-box
- Ogni byte in ingresso viene mappato in un nuovo byte nel seguente modo:
 - I primi quattro bit indicano la riga e i rimanenti 4 bit la colonna
 - I valori di riga e colonna fungono da indici per la S-box

Trasformazione Substitute Bytes (diretta)

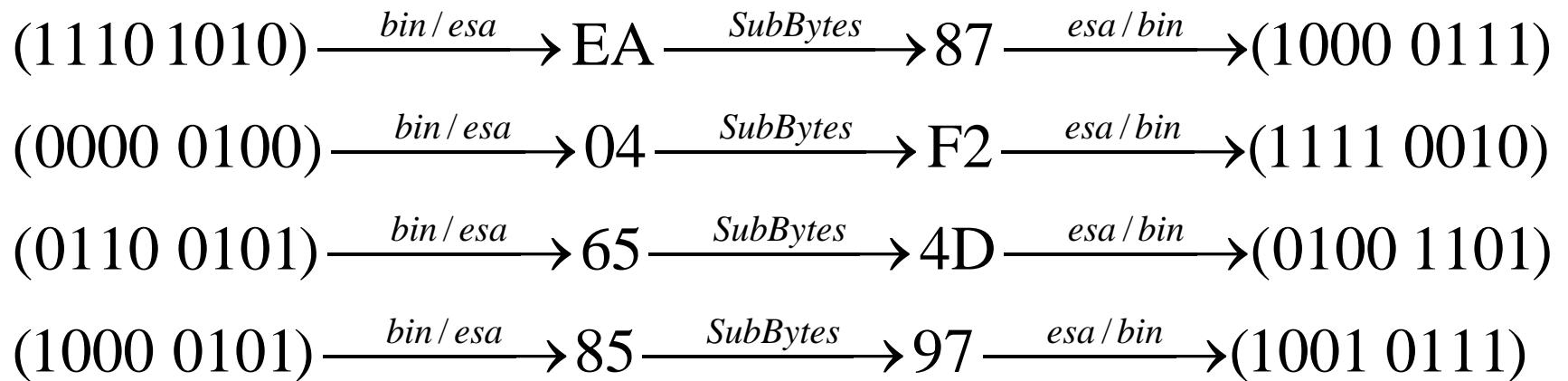
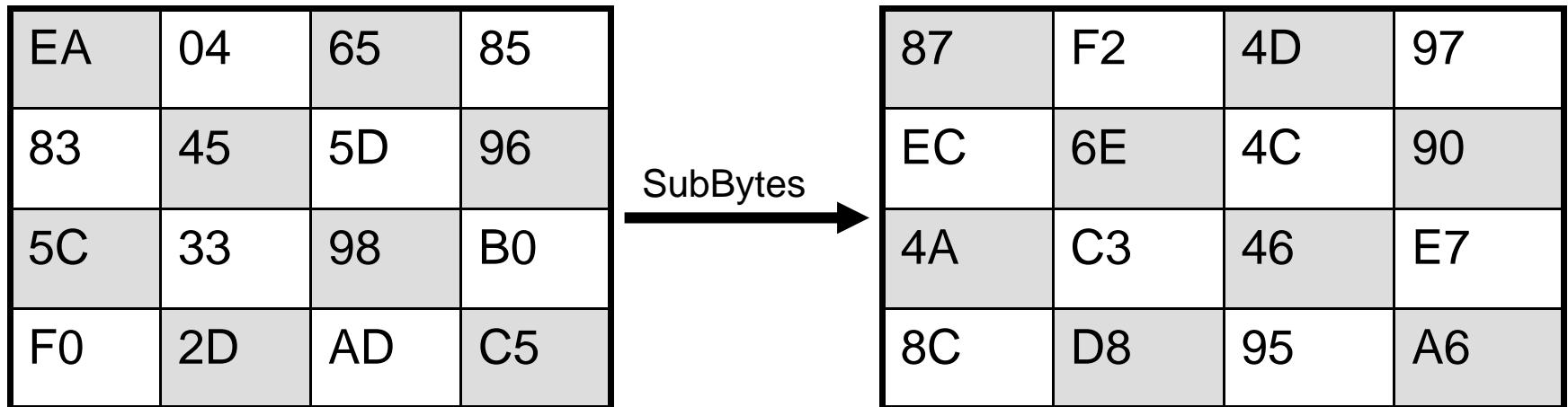


Trasformazione Substitute Bytes (diretta)

A. S-Box

		y																
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
x		0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
		1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
		2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
		3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
		4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
		5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
		6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
		7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
		8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
		9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
		A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	9J	95	E4	79
		B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
		C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
		D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
		E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
		F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Esempio



Progetto della S-box

- Inizializzare la S-box con il valore dei byte in sequenza ascendente. La prima riga contiene {00},{01},...,,{0F}.
- Associate a ciascun byte della S-box il suo inverso moltiplicativo su GF(2⁸), con il polinomio irriducibile $m(x)=x^8+x^4+x^3+x+1$; {00} è mappato su se stesso.
- Considerare che ciascun byte della S-box è costituito da 8 bit (b_7, b_6, \dots, b_0). Applicare la trasformazione:

$$b'_i = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$$

con $\mathbf{c} = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0) = (0,1,1,0,0,0,1,1)$

Progetto della S-box

- Progettata per resistere a tutti gli attacchi noti ad analisi crittografica.
- Bassa correlazione fra i bit in ingresso e in uscita (l'output non è una semplice funzione matematica dell'input).
- La costante additiva **c** è stata scelta in modo da non avere punti fissi ($S\text{-box}(a)=a$) e punti fissi opposti ($S\text{-box}(a)=\text{neg}(a)$).

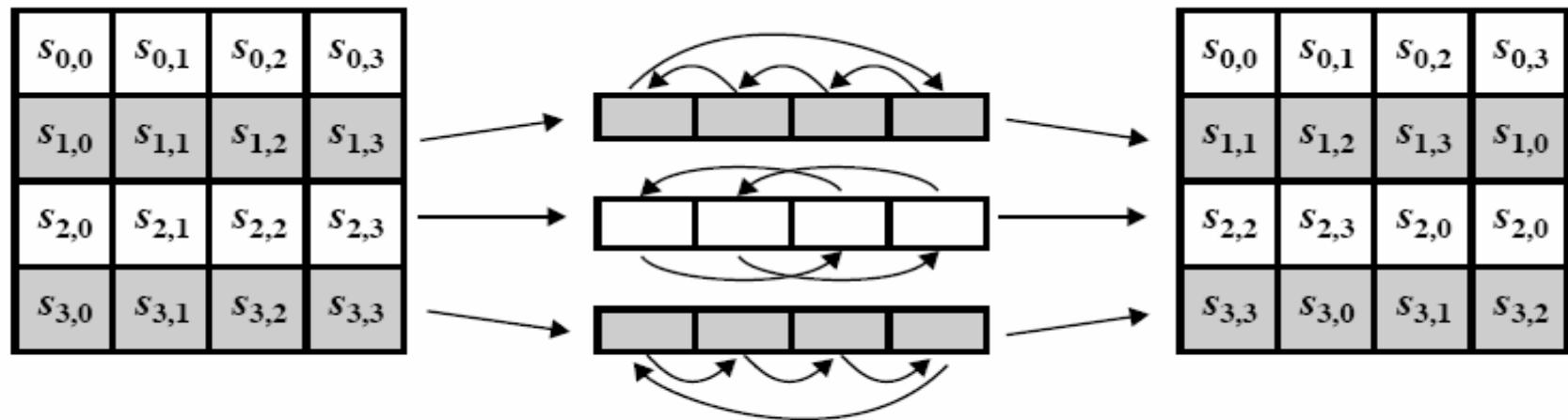
Trasformazione Substitute Bytes (inversa)

B. S-Box inversa

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Trasformazione Shift Rows

- **Diretta:** Riga 1 non viene modificata; Riga 2 scorrimento circolare a sx di un byte; Riga 3 scorrimento circolare a sx di 2 byte; Riga 4 scorrimento circolare a sx di 3 byte.
- **Inversa:** gli scorrimenti sono effettuati verso destra.



(a) Shift row transformation

Esempio

The diagram illustrates the Shift Rows operation on a 4x4 matrix. An arrow labeled "Shift Rows" points from the original matrix on the left to the shifted matrix on the right.

Original Matrix:

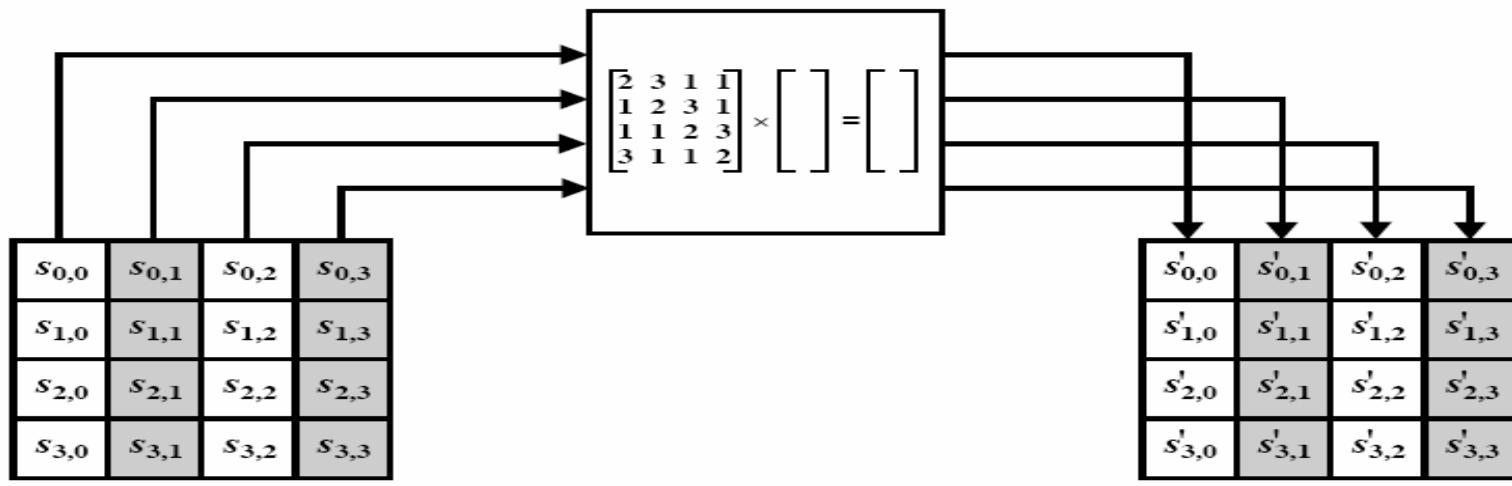
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

Shifted Matrix:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Trasformazione Mix Columns (Diretta)

- Opera su una singola colonna: ciascun byte di una colonna viene mappato in un nuovo valore funzione dei 4 byte presenti nella colonna.
- Somme e moltiplicazioni sono eseguite secondo l'aritmetica di $GF(2^8)$, con $m(x)=x^8+x^4+x^3+x+1$.



(b) Mix column transformation

Aritmetica su GF(2⁸)

- La **somma** di due polinomi in GF(2⁸) corrisponde all'operazione di XOR bit-a-bit.

ESEMPIO

notazione polinomiale :

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$$

notazione binaria :

$$(01010111) \oplus (10000011) = (11010100)$$

notazione esadecimale :

$$\{57\} \oplus \{83\} = \{D4\}$$

Aritmetica su GF(2⁸)

- La **moltiplicazione** su GF(2⁸) può essere difficile da implementare. Tuttavia, utilizzando come polinomio irriducibile $m(x)=x^8+x^4+x^3+x+1$, si ha

$$x^8 \bmod m(x) = [m(x) - x^8] = (x^4 + x^3 + x + 1)$$

da cui

$$f(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

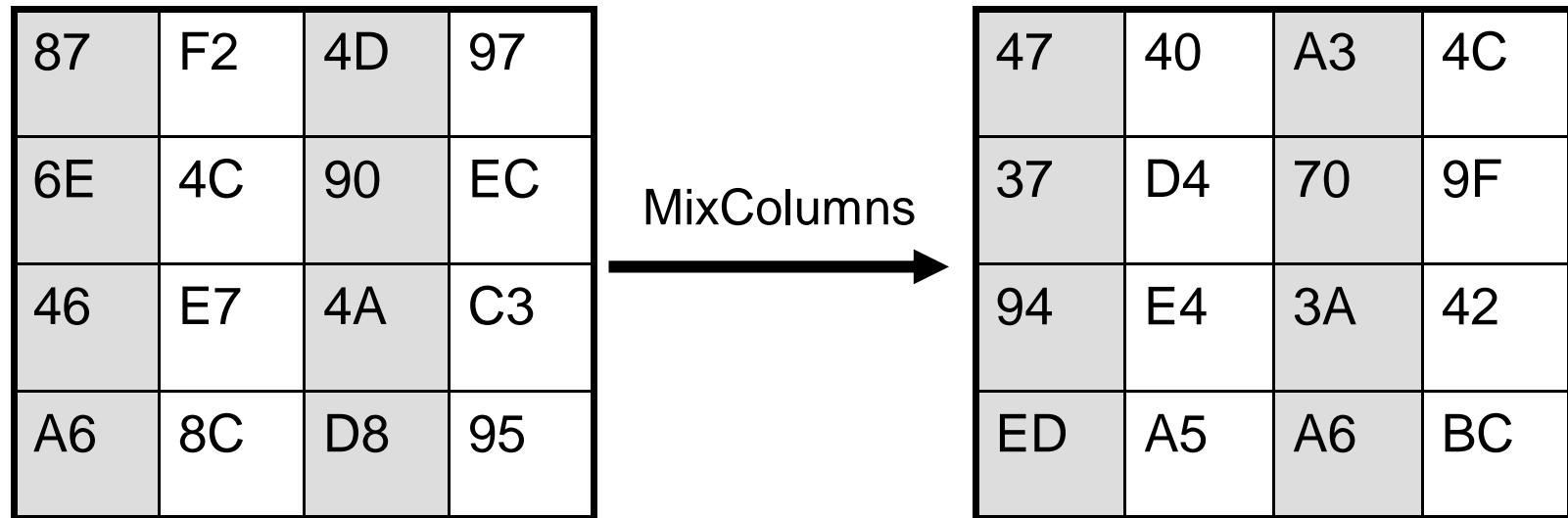
$$x \cdot f(x) = (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x) =$$

$$= \begin{cases} b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x & \text{se } b_7 = 0 \\ (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1) & \text{se } b_7 = 1 \end{cases}$$

In bit:

$$x \cdot f(x) = \begin{cases} (b_6, b_5, b_4, b_3, b_2, b_1, b_0, 0) & \text{se } b_7 = 0 \\ (b_6, b_5, b_4, b_3, b_2, b_1, b_0, 0) \oplus (0, 0, 0, 1, 1, 0, 1, 1) & \text{se } b_7 = 1 \end{cases}$$

Esempio



Esempio (cont.)

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} = \{47\}$$

Infatti :

$$\{87\} \xrightarrow{\text{esa/bin}} (1000\ 0111), \quad \{6E\} \xrightarrow{\text{esa/bin}} (0110\ 1110),$$

$$\{46\} \xrightarrow{\text{esa/bin}} (0100\ 0110), \quad \{A6\} \xrightarrow{\text{esa/bin}} (1010\ 0110),$$

$$\{47\} \xrightarrow{\text{esa/bin}} (0100\ 0111)$$

$$\{02\} \bullet \{87\} \xrightarrow{\text{esa/bin}} (0000\ 1110) \oplus (0001\ 1011) = (0001\ 0101)$$

$$\begin{aligned} \{03\} \bullet \{6E\} \xrightarrow{\text{esa/bin}} & \{6E\} \oplus (\{02\} \bullet \{6E\}) = (0110\ 1110) \oplus (1101\ 1100) \\ & = (1011\ 0010) \end{aligned}$$

$$(0001\ 0101) \oplus (1011\ 0010) \oplus (0100\ 0110) \oplus (1010\ 0110) =$$

$$(0100\ 0111) \xrightarrow{\text{bin/esa}} \{47\}$$

Trasformazione Mix Columns (Inversa)

- Si usa una matrice di trasformazione che è l'inversa moltiplicativa della matrice di trasformazione diretta:

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} = \begin{pmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{pmatrix}$$

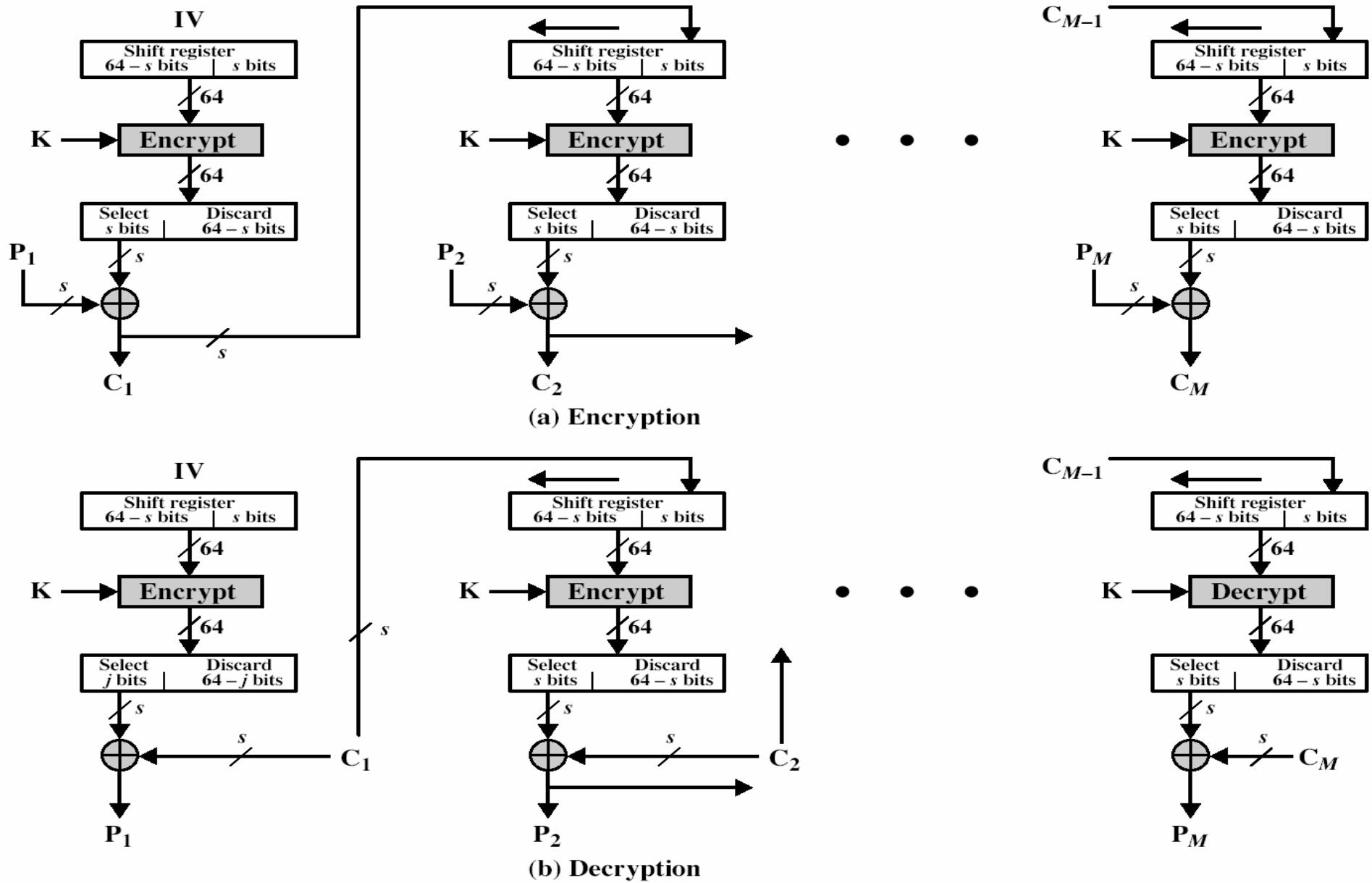
con

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

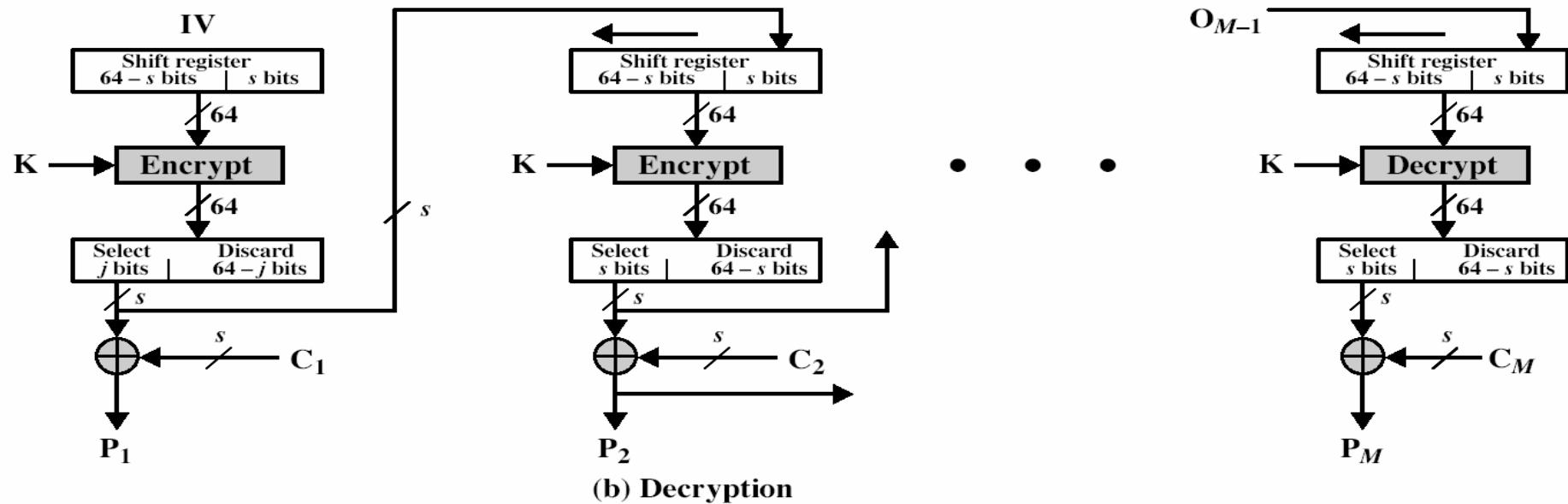
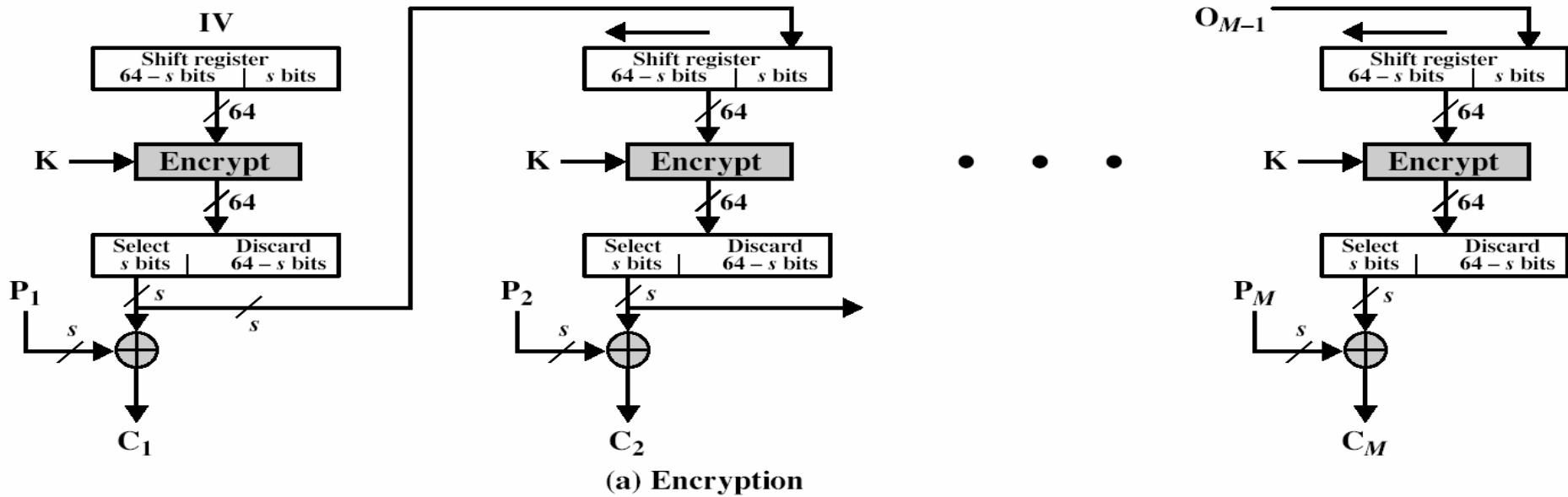
Trasformazione Mix Columns – Commenti generali

- Mix Columns combinata con Shift Rows garantisce che dopo alcune fasi tutti i bit di output dipendano da tutti i bit di input.
- La scelta dei coefficienti della trasformazione diretta {01}, {02} e {03} semplifica l'implementazione:
 - La moltiplicazione richiede al più uno scorrimento e uno XOR.
- La trasformazione inversa è invece più complessa da implementare.
 - Per le modalità Cipher Feedback e Output Feedback si usa solo la crittografia.

Cipher FeedBack (CFB)

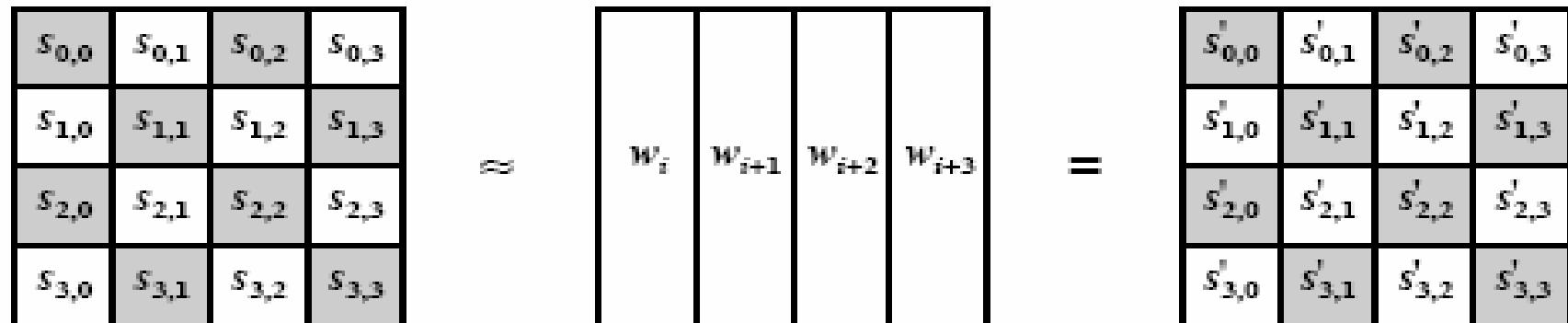


Output FeedBack (OFB)



Trasformazione Add Round Key (Diretta e Inversa)

- XOR della matrice di dati con i 128 bit della chiave di fase.
- Trasformazione molto semplice: sicurezza garantita dalla complessità dell'espansione della chiave e dalla complessità delle altre trasformazioni.



(b) Add Round Key Transformation

Esempio

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC
⊕			
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A
=			
EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

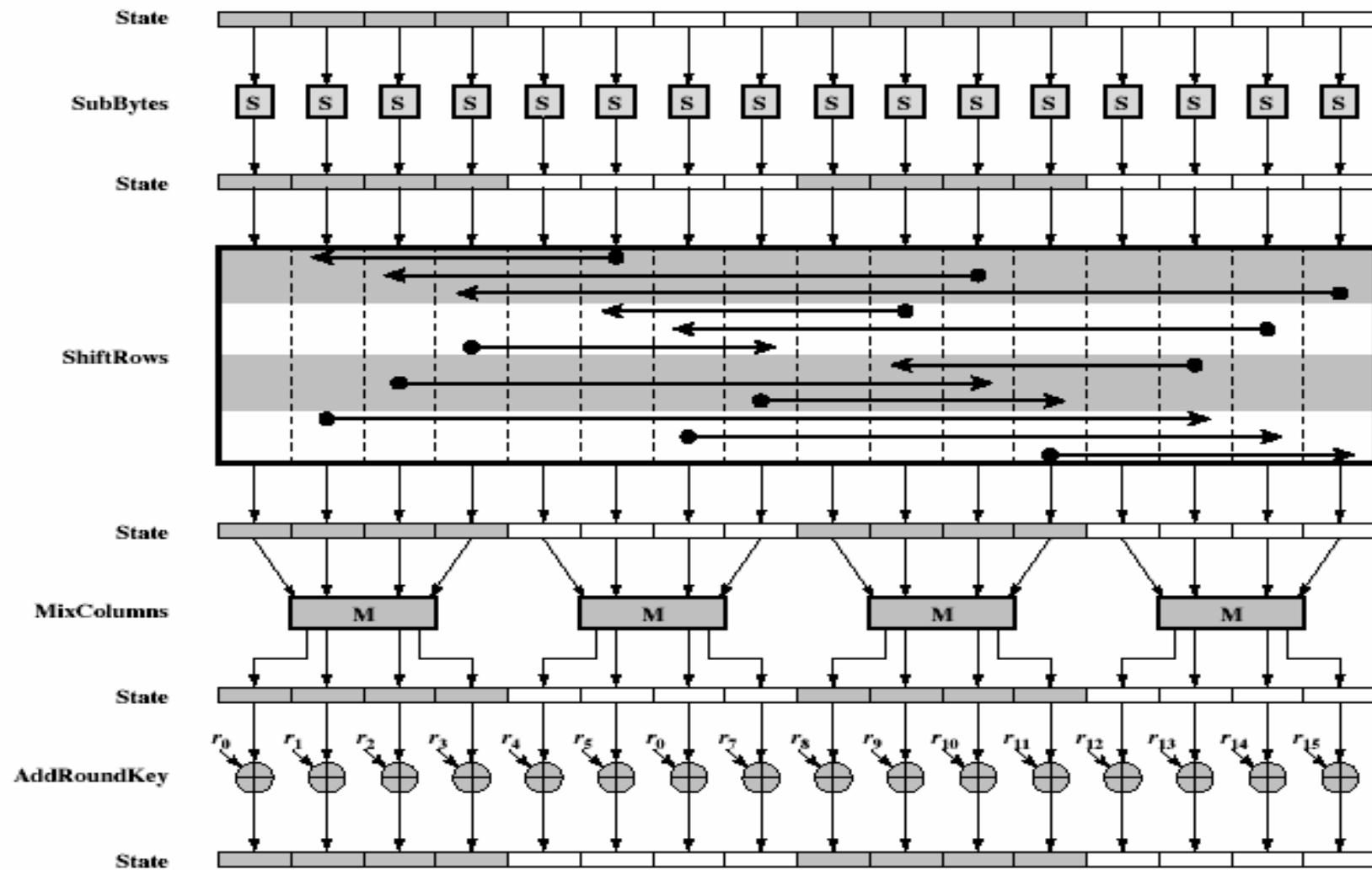
$$47 \xrightarrow{\text{esa / bin}} (0100\ 0111), \quad AC \xrightarrow{\text{esa / bin}} (1010\ 1100)$$

$$(0100\ 0111) \oplus (1010\ 1100) = (1110\ 1011) \xrightarrow{\text{bin / esa}} EB$$

$$40 \xrightarrow{\text{esa / bin}} (0100\ 0000), \quad 19 \xrightarrow{\text{esa / bin}} (0001\ 1001)$$

$$(0100\ 0000) \oplus (0001\ 1001) = (0101\ 1001) \xrightarrow{\text{bin / esa}} 59$$

Schema di una fase di AES



Espansione della chiave

- A partire da una chiave di 4 word (16 byte) viene prodotto un array di 44 word (156 byte).
- L'espansione è descritta dal seguente pseudocodice:

```
KeyExpansion (byte key[16], word w[44])  
{  
    word temp;  
    for (i = 0; i < 4; i ++), w[i] = (key[4i],key[4i+1],key[4i+2],key[4i+3]);  
    for (i = 4; i < 44; i ++)  
    {  
        temp = w[i - 1];  
        if (i mod 4 = 0) temp = SubWord(RotWord(temp)) ⊕ Rcon[i/4];  
        w[i] = w[i - 4] ⊕ temp;  
    }  
}
```

Espansione della chiave

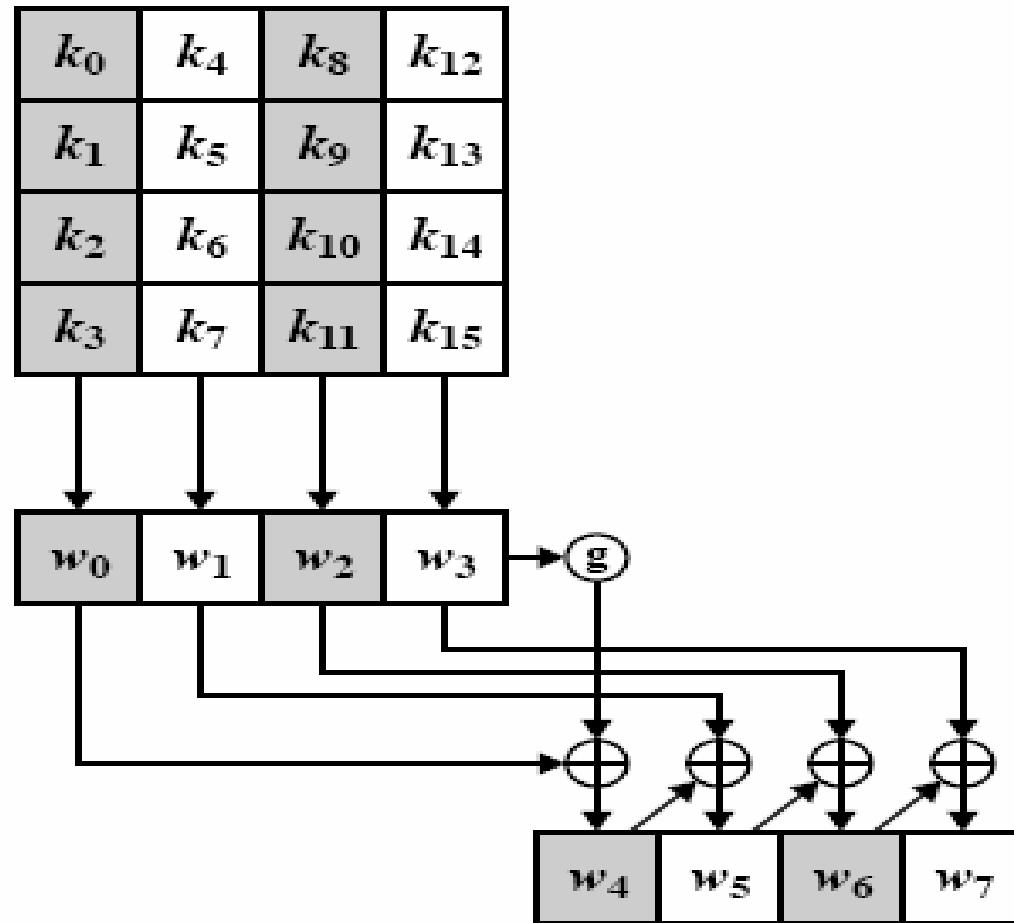
- La chiave viene copiata nelle prime 4 word della chiave espansa.
- Ciascuna word aggiuntiva $w[i]$ dipende da $w[i-1]$ e $w[i-4]$:
 - Se i non è multiplo di 4, viene usata una semplice funzione di XOR;
 - Altrimenti viene usata una funzione più complessa

Espansione della chiave

- **RotWord** → svolge uno scorrimento circolare a sinistra su una word
- **SubWord** → svolge una sostituzione su ciascun byte della word utilizzando la medesima S-box della trasformazione Substitute Bytes diretta.
- **Rcon[j]** → costante di fase. E' una word in cui i tre byte più a destra sono zero. In esadecimale si ha: $Rcon[j] = (RC[j], 0, 0, 0)$, con

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Schema di generazione delle prime 8 word della chiave espansa

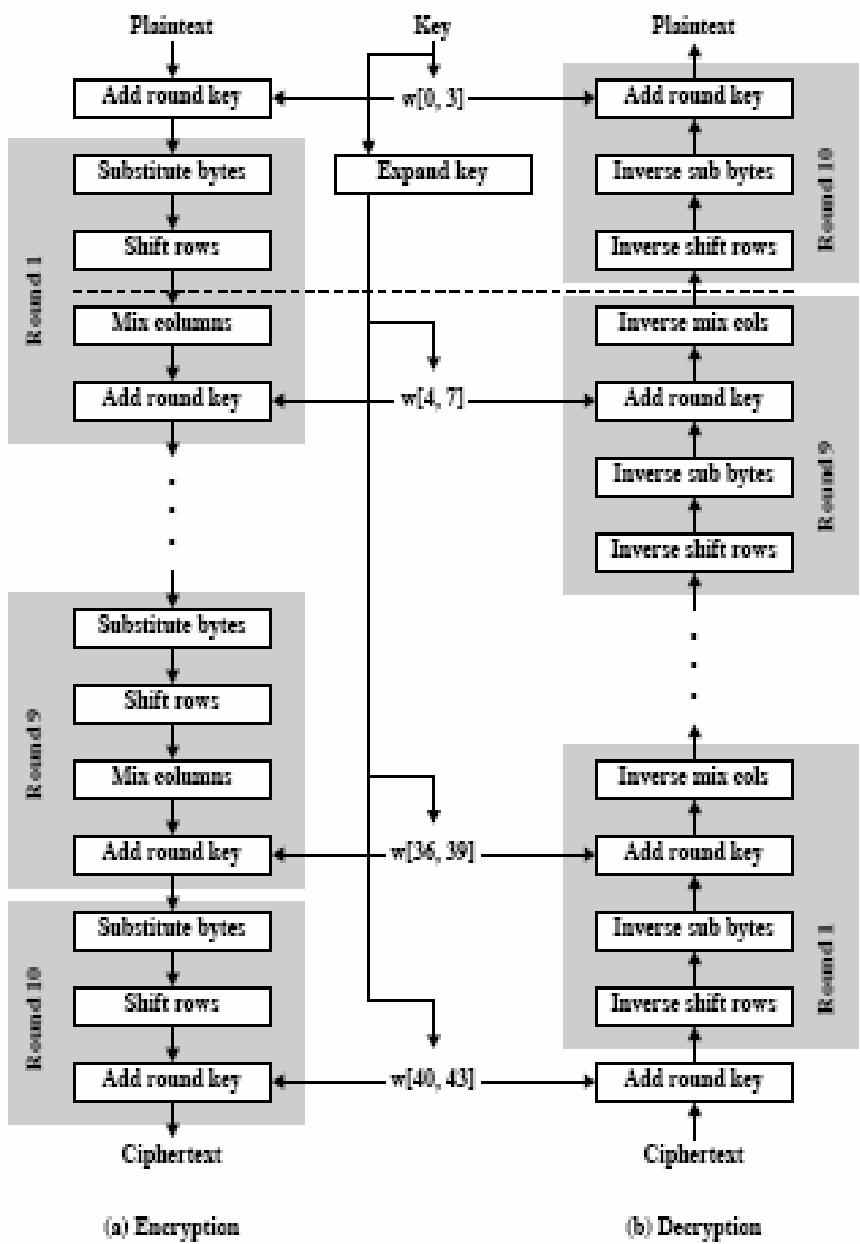


Criteri usati per la generazione della chiave espansa

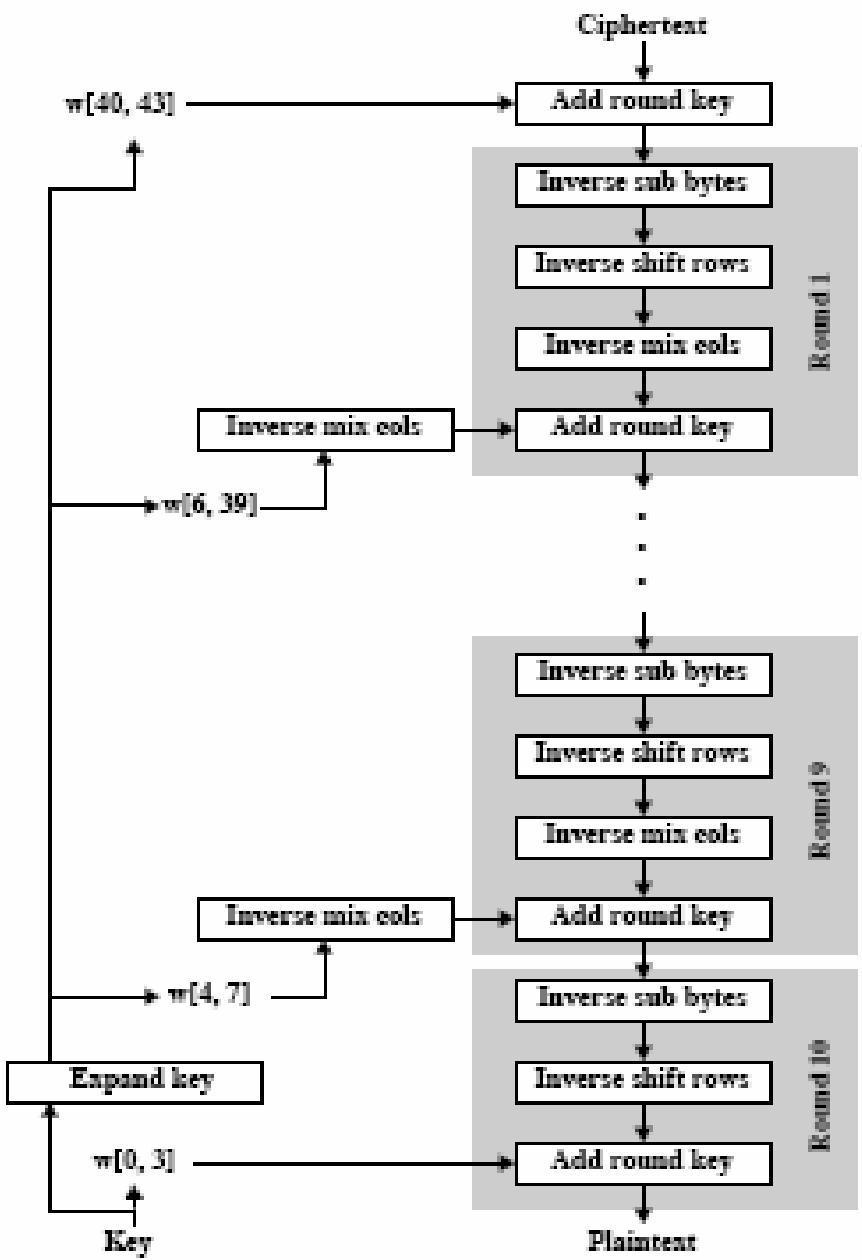
- Resistenza ad attacchi crittografici
- Trasformazione invertibile
- Alta velocità di esecuzione
- Uso di costanti di fasi per eliminare le simmetrie nelle varie fasi
- Ogni bit della chiave crittografica influenza più bit della chiave espansa
- Algoritmo di generazione non lineare.
- Facilità di descrizione

Cifratura inversa equivalente

- Per la crittografia e la decrittografia di AES, la programmazione della chiave è la stessa, ma la sequenza di operazioni da effettuare è diversa.
 - Sono richiesti due moduli software o firmware per le applicazioni che effettuano entrambe le operazioni
- Vi è una versione equivalente dell'algoritmo di decrittografia che usa la stessa sequenza di operazioni di quello di crittografia (invertendole), ma una diversa programmazione della chiave.



(a) Encryption



(b) Decryption

Cifratura inversa equivalente

- **Scambio InvShiftRows e InvSubBytes:** InvShiftRows altera la sequenza dei byte, ma non il loro contenuto; InvSubBytes agisce sul contenuto del byte, indipendentemente dalla sua posizione → Possono essere invertite.
- **Scambio InvMixColumns e AddRoundKey:** si ha

$$\text{InvMixColumns}(S_j \oplus w_j) = \text{InvMixColumns}(S_j) \oplus \text{InvMixColumns}(w_j)$$

infatti

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} s_0 \oplus w_0 \\ s_1 \oplus w_1 \\ s_2 \oplus w_2 \\ s_3 \oplus w_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \oplus \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

Aspetti implementativi

- Implementazione efficiente su CPU a 8 bit (presenti nelle smart card):
 - **Add Round Key**: semplice operazione di XOR
 - **Shift Rows**: operazione di scorrimento dei byte
 - **Substitute Bytes**: opera a livello del byte e richiede una tabella di soli (16x16) 256 byte
 - **Mix Columns**: richiede solo scorrimenti e XOR
- Implementazione efficiente su CPU a 32 bit: tutte le operazioni possono essere espresse sulle word invece che sui singoli byte.

Algoritmo di crittografia **Blowfish**

Caratteristiche di Blowfish

- Sviluppato da Bruce Schneier (1993)
- **Velocità:** su CPU a 32bit può crittografare dati ad una frequenza di 18 cicli di clock per byte (DES, invece, richiede 50 cicli di clock per byte).
- **Compatezza:** può operare con 5KB di memoria.
- **Semplicità:** la struttura è facile da implementare (facilità di analisi).
- **Sicurezza regolabile:** lunghezza della chiave regolabile da 32 a 448 bit.
- **Blocco dati:** 64 bit

Generazione della sottochiave e della S-box

- Una chiave da 32 a 448 bit (ovvero, da 1 a 14 word da 32 bit) viene usata per:
 - generare 18 sottochiavi a 32 bit
 - generare 4 S-box 256x32
- La chiave è conservata in una matrice K:
 $K=[K_1, K_2, \dots, K_j]$ con $1 \leq j \leq 14$.
- Le sottochiavi sono memorizzate in un array P:
 $P=[P_1, P_2, \dots, P_{18}]$

Generazione della sottochiave e della S-box

- Ciascuna delle 4 S-Box è memorizzata in 256 voci da 32 bit:

$$S_{1,0}, S_{1,1}, \dots, S_{1,255}$$

$$S_{2,0}, S_{2,1}, \dots, S_{2,255}$$

$$S_{3,0}, S_{3,1}, \dots, S_{3,255}$$

$$S_{4,0}, S_{4,1}, \dots, S_{4,255}$$

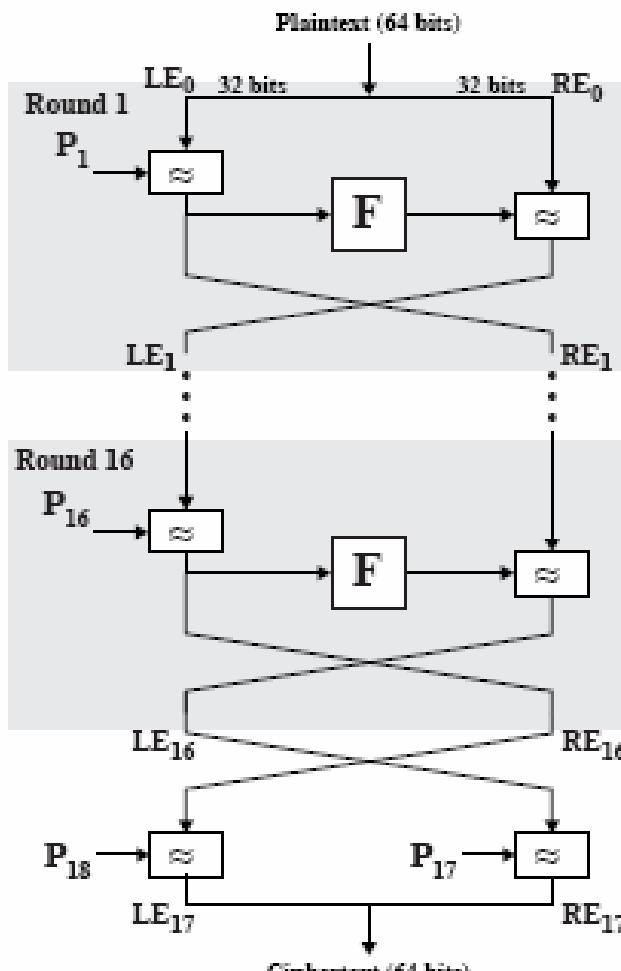
Generazione della sottochiave e della S-box

1. Inizializzare prima l'array P e poi le 4 S-Box utilizzando i bit della parte frazionaria della costante π .
2. Svolgere uno XOR bit-a-bit dell'array P e dell'array K, riutilizzando ciclicamente le word dell'array K. Per esempio, per la chiave di lunghezza massima (14 word da 32 bit): $P_1 = P_1 \oplus K_1$, $P_2 = P_2 \oplus K_2, \dots, P_{14} = P_{14} \oplus K_{14}$, $P_{15} = P_{15} \oplus K_1, \dots, P_{18} = P_{18} \oplus K_4$.
3. Crittografare un blocco di 64 bit nulli utilizzando P e le 4 S-Box correnti. Sostituire P_1 e P_2 con l'output della crittografia.
4. Crittografare l'output del passo 3 utilizzando P e le 4 S-Box correnti e sostituire P_3 e P_4 con l'output della crittografia.
5. Continuare questa operazione per aggiornare tutte le word di P e tutte le word delle 4 S-Box.

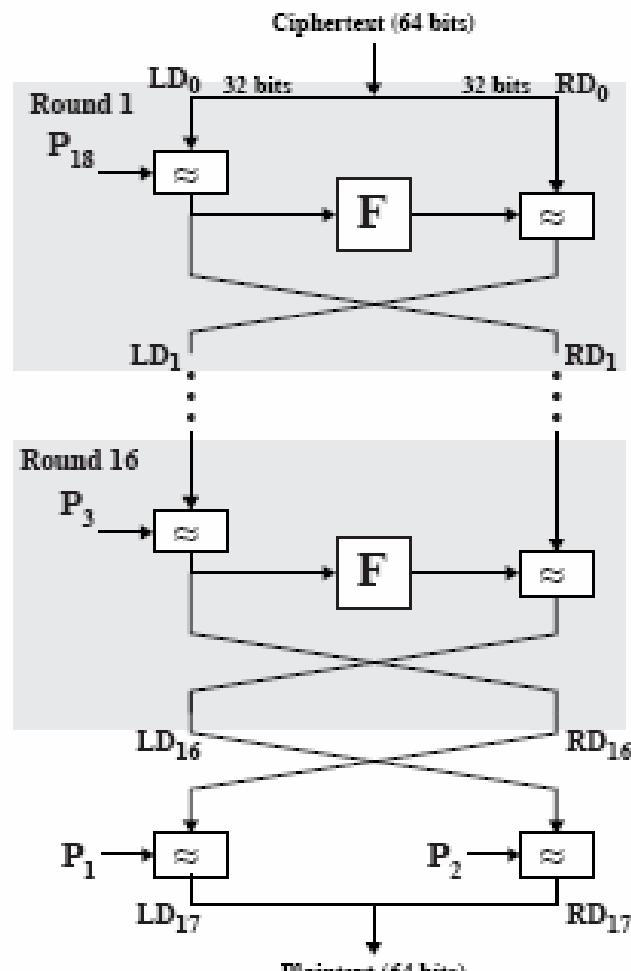
Commenti

- Per produrre l'array finale P e le 4 S-Box finali sono necessarie 521 esecuzioni consecutive dell'algoritmo di crittografia.
 - Blowfish non è adatto per applicazioni in cui la chiave segreta cambia spesso
 - Per memorizzare la matrice P e le 4 S-Box sono necessari più di 4 KB di memoria
 - Resistente contro attacchi a forza bruta (fino ad ora la sicurezza di Blowfish è ritenuta inattaccabile!)

Crittografia e Decrittografia



(a) Encryption



(b) Decryption

Figure 6.3 Blowfish Encryption and Decryption

Funzione F (S-Box)

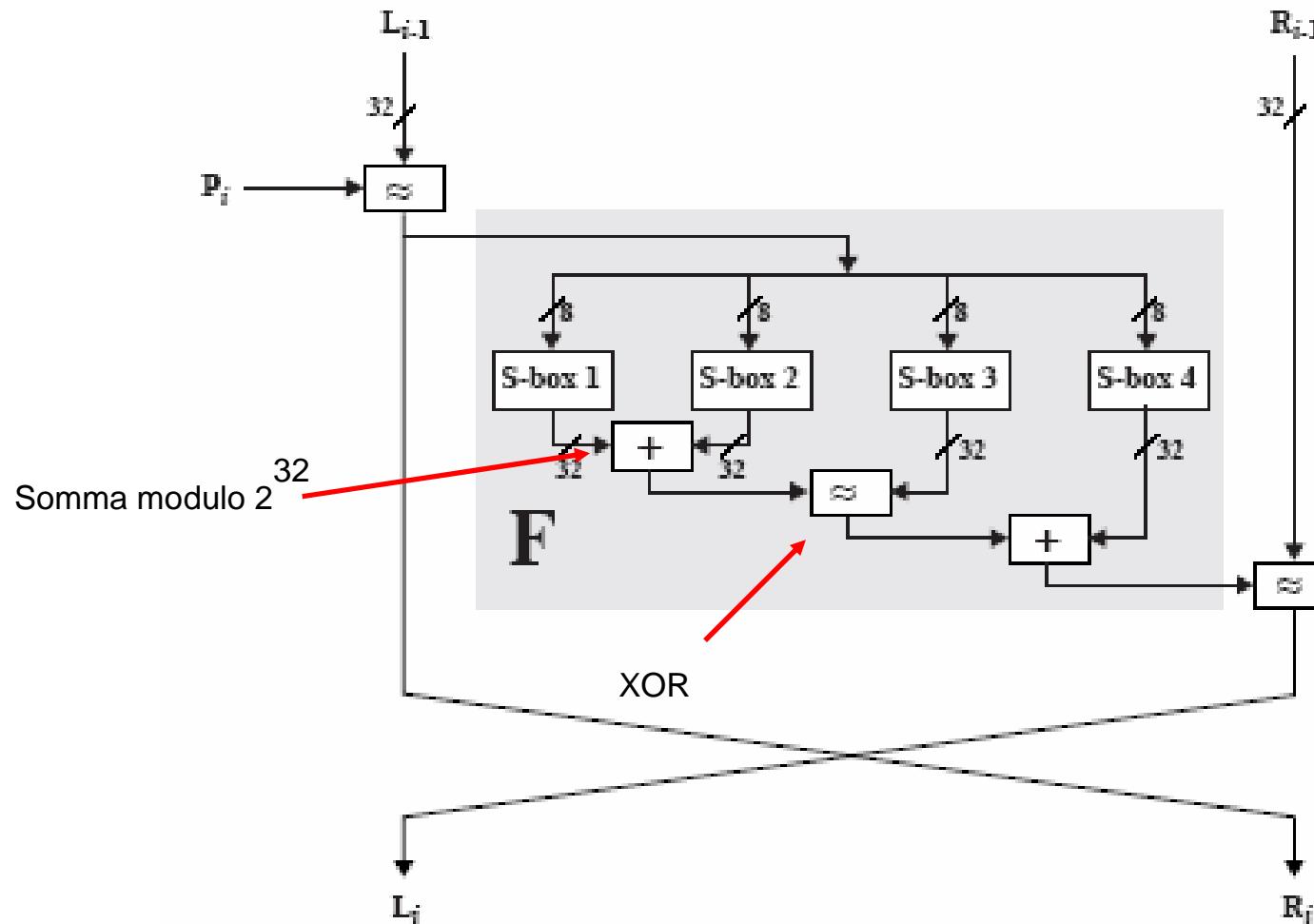


Figure 6.4 Detail of Single Blowfish Round

Funzione F (S-Box)

- Sono usate due operazioni primitive:
 - **Somma di word**, rappresentata con $+$, eseguita modulo 2^{32}
 - **XOR bit-a-bit**, rappresentato con \oplus
- L'input di 32 bit di F viene diviso in 4 byte. Se questi byte vengono indicati con a, b, c, d , si ha

$$F[a,b,c,d] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$$

Blowfish: descrizione

- La generazione di una chiave richiede 521 applicazioni dell'algoritmo: difficoltà dell'attacco a forza bruta
- La funzione F introduce un forte effetto valanga
- Le S-box dipendono dalla chiave
- La funzione F non dipende dalla fase

Crittografia a chiave pubblica

Cenni alla teoria dei numeri

Numeri Primi

- I numeri primi hanno come divisori solo se stessi e l'unità
 - Non possono essere ottenuti dal prodotto di altri numeri
 - eg. 2,3,5,7 sono primi, 4,6,8,9,10 non lo sono
 - I numeri primi svolgono un ruolo fondamentale nella teoria dei numeri
 - Ecco l'elenco dei numeri primi inferiori a 200:

Fattorizzazione in numeri primi

- Ogni intero può essere diviso in fattori in modo univoco
- N.B.: Fattorizzare un numero è alquanto più complicato che ottenere il numero stesso mediante moltiplicazione dei fattori
- Esempi di fattorizzazione:

$$91 = 7 \times 13 \quad ; \quad 3600 = 2^4 \times 3^2 \times 5^2$$

Fattorizzazione in numeri primi

- In generale, dato un intero a , vale la relazione

$$a = p_1^{a_1} p_2^{a_2} p_3^{a_3} \dots$$

ossia

$$a = \prod_{p \in P} p^{a_p}, \quad a_p \geq 0$$

ove P è l'insieme dei numeri primi

Fattorizzazione in numeri primi

- Il valore di un intero positivo lo si può specificare semplicemente indicando i valori degli esponenti non nulli, ossia:

$$12=\{a_2=2, a_3=1\}, \quad 18=\{a_2=1, a_3=2\}$$

- La moltiplicazione di due numeri equivale alla somma degli esponenti:
- $K=mn \rightarrow k_p=m_p+n_p$, per ogni $p \in P$

Numeri Relativamente primi & GCD

- I numeri a, b sono **relativamente primi** se, a parte l'unità, non hanno divisori comuni, ovvero $\text{GCD}(a, b)=1$
 - e.g. 8 & 15 sono relativamente primi
- Il GCD può essere calcolato dalla scomposizione in fattori primi
 - eg. $300=2^1\times3^1\times5^2$ $18=2^1\times3^2$, da cui:
 $\text{GCD}(18, 300)=2^1\times3^1\times5^0=6$

Il Piccolo Teorema di Fermat



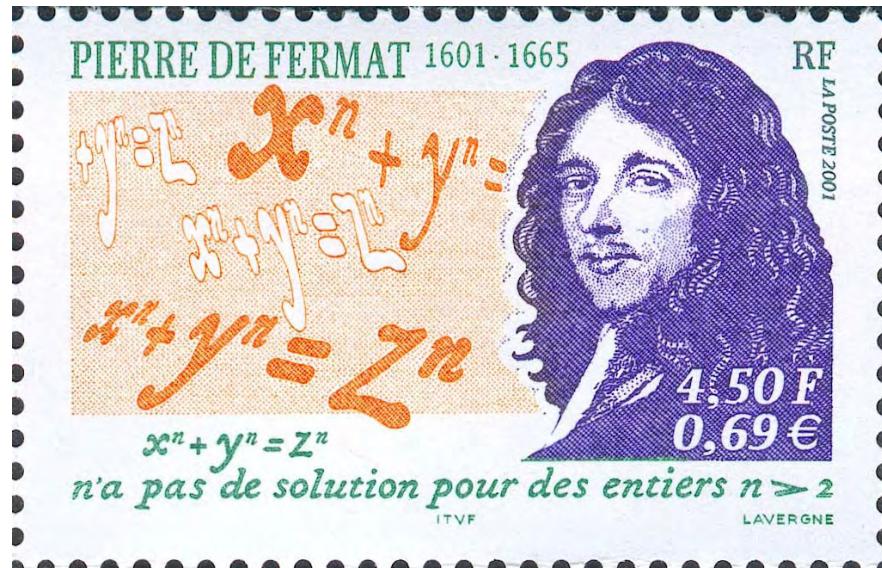
Pierre de Fermat

Born: 17.09.1601 in Beaumont-de-Lomagne

Died: 12.01.1665 in Castres

Dato un numero primo p e un intero a tale che $\gcd(a,p)=1$, è:

$$a^{p-1} \bmod p = 1$$



Il Piccolo Teorema di Fermat: Prova (1/2)

- Poichè p è primo, $\mathbb{Z}_p = \{ 0, \dots, p-1 \}$ è tale che moltiplicando modulo p gli elementi di \mathbb{Z}_p per a , si riottengono gli stessi elementi di \mathbb{Z}_p in ordine diverso;
- Moltiplicando gli elementi di entrambi gli insiemi (ad eccezione dello 0) si ottiene
$$1 \times 2 \times \dots \times (p-1) \text{ mod } p = (p-1)! \text{ mod } p$$
$$a \times 2a \times \dots \times (p-1)a \text{ mod } p = a^{p-1} (p-1)! \text{ mod } p$$

Il Piccolo Teorema di Fermat: Prova (2/2)

- Pertanto si ha
 - $(p-1)!a^{p-1} \text{ mod } p = (p-1)! \text{ mod } p$
- Semplificando $(p-1)!$ (lo si può fare, perché??) si ha la tesi.

Una formulazione equivalente del teorema è ovviamente

$$a^p \text{ mod } p = a$$

La Funzione Toziente di Eulero $\Phi(n)$

- Dato un intero positivo n, la funzione toziente di Eulero $\Phi(n)$ è pari al numero di interi positivi minori di n e primi relativi con n, ovvero

$$\Phi(n)=\text{card}(\{x \in \mathbb{Z}_n : \gcd(x,n)=1\})$$

- Esempi:
 $\Phi(3)=2$, $\Phi(5)=4$, $\Phi(6)=2$.
- Nota: $\Phi(p)=p-1$, per ogni p numero primo

La Funzione Toziente di Eulero

Table 8.2 Some Values of Euler's Totient Function $\phi(n)$

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

La Funzione Toziente di Eulero $\Phi(n)$

- Il calcolo di $\Phi(n)$ richiede in genere la verifica e la conta esaustiva degli elementi di Z_n relativamente primi con n
- Valgono le seguenti eccezioni
 - Se p è primo $\Phi(p) = p-1$
 - Se p e q sono primi $\Phi(pq) = (p-1)(q-1)$
- eg.
 - $\Phi(37) = 36$
 - $\Phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$

Teorema di Eulero

- È una generalizzazione del teorema di Fermat

Se a e n sono primi relativi, allora

$$a^{\Phi(n)} \bmod n = 1$$

- Esempi

- $a=3; n=10; \Phi(10)=4;$
- quindi $3^4 = 81 = 1 \bmod 10$
- $a=2; n=11; \Phi(11)=10;$
- hence $2^{10} = 1024 = 1 \bmod 11$

Teorema di Eulero: Prova (1/3)

- Se n è primo, $\Phi(n)=n-1$ e la prova è banale (th. di Fermat)
- Se n non è primo, si considerino i $\Phi(n)$ interi positivi minori di n e relativamente primi con n : $R=\{x_1, \dots, x_{\Phi(n)}\}$
- Si consideri ora l'insieme S dato da

$$S=\{ax_1 \bmod n, \dots, ax_{\Phi(n)} \bmod n\}$$

Teorema di Eulero: Prova (2/3)

- L'insieme S coincide con R in quanto
 - a e x_i sono entrambi primi relativi con n , e quindi lo è anche ax_i . Pertanto gli elementi di S sono tutti interi minori di n e primi relativi di n
 - in S non vi sono duplicati in quanto se $ax_i \pmod n = ax_j \pmod n$, allora $x_i = x_j$.

Teorema di Eulero: Prova (3/3)

Ma allora, se $R=S$, si ha

$$\prod_{i=1}^{\Phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\Phi(n)} x_i \bmod n$$

da cui segue banalmente la tesi.

Una forma alternativa del teorema è

$$a^{\Phi(n)+1} \bmod n = a$$

Perchè studiamo questa roba??

- Siano p e q due primi, sia $n=pq$ e sia $0 < m < n$
- Vale la relazione

$$\underline{m^{\Phi(n)+1} = m^{(p-1)(q-1)+1} \bmod n = m \bmod n}$$

Se m ed n sono primi relativi, la relazione è vera per il teorema di Eulero. In realtà, tale relazione si dimostra essere valida anche se m ed n non sono primi relativi

- Tale relazione è alla base del funzionamento dell'algoritmo RSA di crittografia a chiave pubblica

Test di primalità

- Nella crittografia, c'è bisogno di procedure per generare numeri primi grandi
- Dato un numero generato in maniera pseudocasuale, bisogna stabilire se è un numero primo
- Una procedura sistematica consiste nel provare a dividere per tutti i numeri primi minori della radice quadrata del numero sotto test
- Chiaramente, al crescere di n tale approccio è inutilizzabile

Test di primalità

- Si ricorre quindi ad un approccio statistico, ovvero si usano test che garantiscono che un numero sia primo “con una certa affidabilità,” ovvero in senso probabilistico
- Tali test si basano sulla verifica di una proprietà
 - Soddisfatta da tutti i numeri primi
 - ... **Ma usualmente soddisfatta anche da qualche numero composto**

Test di Fermat

- Il teorema di Fermat afferma che se n è primo, per ogni $a < n$ tale che $\text{gcd}(a,n)=1$ si ha:

$$a^{n-1} \equiv 1 \pmod{n}$$

Esempio: consideriamo $341 = 11 \times 31$

$$2^{340} \equiv 1 \pmod{341}$$

$$3^{340} \equiv 56 \pmod{341} \Rightarrow 341 \text{ è composto}$$

- N.B.: Il test di Fermat permette di stabilire con certezza che un numero è composto, ma non può provare che esso sia primo
- Inoltre, il test di Fermat prova che un numero è composto, ma non fornisce la sua scomposizione in fattori primi

I numeri di Carmichael

- Consideriamo il numero $561=3 \times 11 \times 17$
- E' possibile verificare che, comunque si scelga $a < 561$ e primo con 561, il test di Fermat non riesce a dimostrare la non primalità di 561
- Sfortunatamente, esistono infiniti numeri di tal tipo
- Essi sono detti *numeri di Carmichael*

I numeri di Carmichael

- 561 è il più piccolo numero di Carmichael
- E' stato dimostrato che esistono infiniti numeri di Carmichael
- 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, ... sono tutti numeri di Carmichael
- Di conseguenza, il test di primalità di Fermat non è affidabile!

Test di Miller Rabin

- E' basato sul teorema di Fermat
- Diamo l'algoritmo senza dimostrazione

TEST (n) is:

1. Find integers k, q , $k > 0$, q odd, so that $(n-1) = 2^k q$
2. Select a random integer a , $1 < a < n-1$
3. **if** $a^q \text{ mod } n = 1$ **then** return ("maybe prime"), STOP;
4. **for** $j = 0$ **to** $k-1$ **do**
5. **if** $(a^{2^j q} \text{ mod } n = n-1)$
 then return(" maybe prime "), STOP
6. **return** ("composite"), STOP

Test di Miller Rabin

Esempio: $n=29$; $n-1=28=2^2 \times 7$; si prenda $a=10$;
 $10^7 \bmod 29=17$ che è diverso da 1 e -1

$10^{7 \times 2} \bmod 29=28 \Rightarrow \text{MAYBE PRIME}$

Si riprovi con $a=2$

$2^7 \bmod 29=12$ che è diverso da 1 e -1

$2^{7 \times 2} \bmod 29=28 \Rightarrow \text{MAYBE PRIME}$

Di fatto, il risultato “maybe prime” si ottiene con tutti gli interi a compresi tra 1 e 28.

Test di Miller Rabin

Esempio: $n=221=13 \times 17$; $n-1=220=2^2 \times 55$; si prenda $a=5$:

$5^{55} \text{ mod } 221=112$ che è diverso da 1 e -1

$5^{55 \times 2} \text{ mod } 221=168 \Rightarrow \text{COMPOSITE}$

Se avessimo scelto $a=21$ si sarebbe avuto
 $21^{55} \text{ mod } 221=200$ che è diverso da 1 e -1

$21^{55 \times 2} \text{ mod } 221=220 \Rightarrow \text{MAYBE PRIME}$

Di fatto, il risultato “maybe prime” si ottiene con 6 dei 220 interi compresi tra 1 e 220.

Considerazioni Probabilistiche

- Se l'esito del test di Miller-Rabin è “composite” il numero è **sicuramente** non primo
- In caso contrario, **potrebbe** essere primo
- È stato mostrato che, se n è composto, scegliendo a caso la base a, la probabilità che il test dia come output “maybe prime” è $< \frac{1}{4}$
- Ripetendo il test t volte con basi a scelte a caso ho che la probabilità che il test dia un output “maybe prime” in presenza di un numero composto è 4^{-t}
- Un numero composto viene identificato quindi con probabilità $1-4^{-t}$
 - E.g., per t=10 tale probabilità è > 0.99999

Distribuzione dei numeri primi

- DOMANDA: Generato un numero dispari n grande, con che probabilità questo è un numero primo???
- Il “teorema dei numeri primi” stabilisce che, per n grande, i numeri primi sono spaziati mediamente ogni $\ln(n)$ interi
- Poichè non si considerano i numeri pari e quelli che terminano per 5, in pratica c’è bisogno in media di $0.4 \ln(n)$ tentativi per poter identificare un numero primo

Distribuzione dei numeri primi

- Nota però che questi sono risultati validi in media....

ESEMPI:

- Gli interi consecutivi $10^{12}+61$ e $10^{12}+63$ sono entrambi primi
- I numeri $1001!+2, 1001!+3, \dots 1001!+1001$ è una sequenza di mille interi consecutivi e non primi

Il Teorema Cinese del Resto

- È utilizzato per velocizzare i calcoli dell'aritmetica modulare
- Si supponga di dover lavorare modulo un certo numero M che è a sua volta prodotto di numeri a coppia relativamente primi
 - eg. mod $M = m_1 m_2 \dots m_k$
- Il Teorema Cinese del Resto ci permette di lavorare con singolarmente modulo ciascuno dei fattori m_i
- Questo permette di risparmiare notevolmente in termini di risorse computazionali

Il Teorema Cinese del Resto

- Il teorema afferma che è possibile ricostruire gli interi di un determinato intervallo a partire dai loro resti modulo una coppia di numeri relativamente primi
- ESEMPIO: Gli elementi in \mathbb{Z}_{10} possono essere rappresentati dai loro resti modulo 2 e 5

Il Teorema Cinese del Resto

- Sia $M = m_1 m_2 \dots m_k$, con $\gcd(m_i, m_j) = 1$ per ogni $i \neq j$
- Per ogni intero $A \in Z_M$, posto $a_i = A \bmod m_i$, la corrispondenza $A \leftrightarrow (a_1, a_2, \dots, a_k)$ è biunivoca

Si tratta in realtà di un mappaggio da Z_M a $Z_{m_1} \times Z_{m_2} \times \dots \times Z_{m_k}$

Il Teorema Cinese del Resto

- Si supponga di voler calcolare $(A \text{ mod } M)$
- Dapprima si calcolano separatamente gli $a_i = (A \text{ mod } m_i)$ e poi si combinano insieme secondo le formule:

$$c_i = M_i^{-1} \mod m_i \quad \text{per } 1 \leq i \leq k$$

$$A = \left(\sum_{i=1}^k a_i c_i \right) \mod M$$

ove $M_i = M/m_i$

Il Teorema Cinese del Resto

ESEMPIO:

$$M=1813=37 \times 49, A=973$$

$$a_1=973 \bmod 37=11, a_2=973 \bmod 49=42$$

$$M_1=49, M_2= 37$$

$$(49)^{-1} \bmod 37=34, (37)^{-1} \bmod 49=4$$

$$c_1 = 49 \times 34=1666, c_2 = 4 \times 37=148$$

$$A=(1666 \times 11 + 148 \times 42) \bmod M=973$$

Radici Primitive

- Sappiamo dal teorema di Eulero che per a ed n primi relativi si ha $a^{\Phi(n)} \text{mod } n=1$
- Si consideri l'equazione in m: $a^m \text{mod } n=1$, con $\gcd(a, n)=1$
 - Una soluzione è certamente $m = \Phi(n)$, ma può esserci una soluzione minore di $\Phi(n)$
 - Le potenze si ripetono poi ciclicamente
- Se la più piccola soluzione è $m = \Phi(n)$ allora a è detto una **radice primitiva** di n
- Se n è primo, allora le potenze successive di a generano Z_p ($= Z_n$)

Radici Primitive

ESEMPIO: Potenze di 7 modulo 19

$$7^1 \bmod 19 = 7$$

$$7^2 \bmod 19 = 11$$

$$7^3 \bmod 19 = 1$$

$$7^4 \bmod 19 = 7$$

$$7^5 \bmod 19 = 11$$

.....

Radici Primitive

- Non tutti gli interi hanno radici primitive
- Gli unici interi con radici primitive sono nella forma $2, 4, p^\alpha$ e $2p^\alpha$.

ESEMPIO:

Le radici primitive di 19 sono 2, 3, 10, 13, 14 e 15

Logaritmi Discreti

- Abbiamo trattato dell'esponenziazione modulare
- La funzione inversa dell'esponenziazione modulare è il cosiddetto **logaritmo discreto** di un numero modulo p
- Il logaritmo di b in base a e modulo p è quell'intero x tale che $a^x \equiv b \pmod{p}$
- Si usa la notazione $x = \log_a b \pmod{p}$ o $x = \text{ind}_{a,p}(b)$

Logaritmi Discreti

- In generale, non è garantita l'esistenza del logaritmo discreto
- Se a è una radice primitiva di m , allora il logaritmo in base a e modulo m esiste sempre
 - $x = \log_3 4 \text{ mod } 13$ (ovvero trova x tale che $3^x \equiv 4 \pmod{13}$) non ha soluzione
 - $x = \log_2 3 \text{ mod } 13 = 4$ per ricerca esaustiva
- N.B.: Mentre l'elevazione a potenza è alquanto semplice, il calcolo del logaritmo discreto è unanimemente riconosciuto non fattibile quando m diventa grande

Sommario

- Ieri abbiamo parlato di:
 - Numeri primi
 - I Teoremi di Fermat e di Eulero
 - Test di primalità (statistici)
 - Il Teorema Cinese del Resto
 - Logaritmi Discreti

Richiami Lezione Precedente

- Il Piccolo Teorema di Fermat

Dato un numero primo p e un intero a tale che $\gcd(a,p)=1$, è:

$$a^{p-1} \bmod p = 1$$

Richiami Lezione Precedente

- Il teorema di Eulero

Se a e n sono primi relativi, allora

$$a^{\Phi(n)} \bmod n = 1$$

Richiami Lezione Precedente

- I Test di primalità
 - Test di Fermat
 - Numeri di Carmichael
 - Test di Miller-Rabin
- Il Teorema cinese del resto

Richiami Lezione Precedente

- Radici primitive:
a è detta radice primitiva di n se
l'equazione $a^m \text{ mod } n=1$, ammette come
soluzione m più piccola $m=\Phi(n)$.

Se n è primo, $a^m \text{ mod } n$ genera, al variare
di m, tutti gli elementi di Z_n ad eccezione
dello 0

Richiami Lezione Precedente

- Logaritmi discreti
 - Il logaritmo di b in base a e modulo p è quell'intero x tale che $a^x \equiv b \pmod{p}$
 - In generale, non è garantita l'esistenza del logaritmo discreto
 - Se a è una radice primitiva di m, allora il logaritmo in base a e modulo m esiste sempre
 - Il calcolo del logaritmo discreto di numeri grandi è complicato!!

Oggi parleremo di....

- Crittografia a chiave pubblica
 - L'algoritmo RSA
- Procedure per lo scambio delle chiavi

Crittografia a singola chiave

- Come detto più volte, la **crittografia a chiave segreta** usa un'unica chiave
- La chiave è condivisa da trasmittitore e ricevitore
- Se la chiave è violata, la comunicazione non è più sicura
- La crittografia è simmetrica, ovvero i due attori della comunicazione sono paritetici
- Quindi il ricevitore potrebbe coniare un messaggio, crittografarlo e sostenere che tale messaggio sia stato inviato dal trasmittitore!!

Crittografia a chiave pubblica

- Probabilmente, trattasi della scoperta più importante nella storia millenaria della crittografia
- utilizza **due** chiavi: una è pubblica e una è privata
- È **asimmetrica** poiché i soggetti della comunicazione non sono uguali
- Si basa su concetti avanzati della teoria dei numeri
- Tuttavia, essa complementa e non rimpiazza l'uso della crittografia simmetrica

Crittografia a chiave pubblica

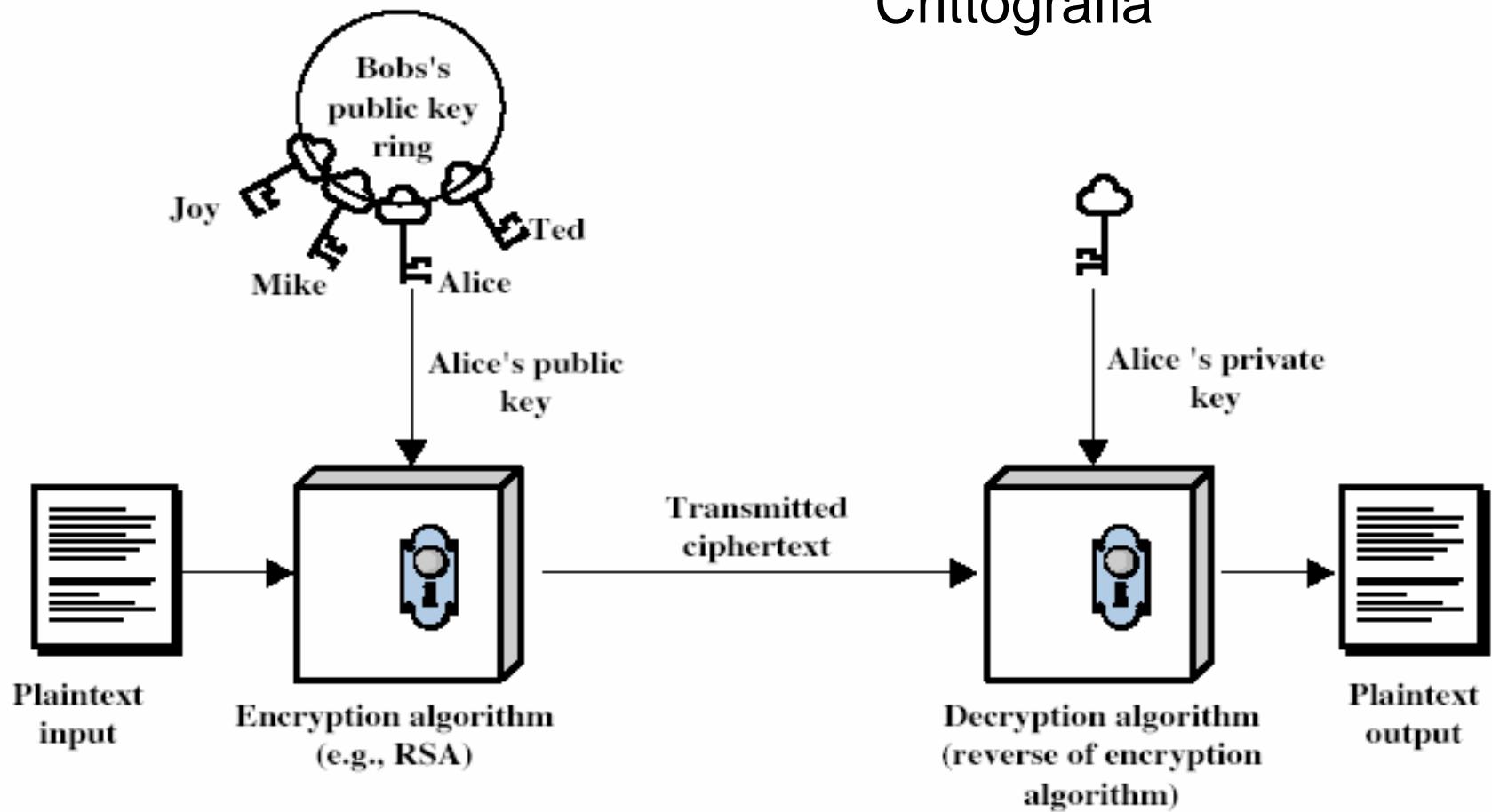
- La crittografia a **chiave pubblica/a due chiavi/asimmetrica** si basa sull'uso di **due chiavi**:
 - una **chiave pubblica**, che è pubblicamente nota, ed è usata per **criptare messaggi e verificare le firme**
 - una **chiave privata**, nota solo a soggetto della comunicazione, usata per **decriptare i messaggi e firmare i messaggi**

Crittografia a chiave pubblica

- è **asimmetrica** perchè
 - Coloro che criptano i messaggi o verificano le firme **non possono** decriptare i messaggi o creare le firme

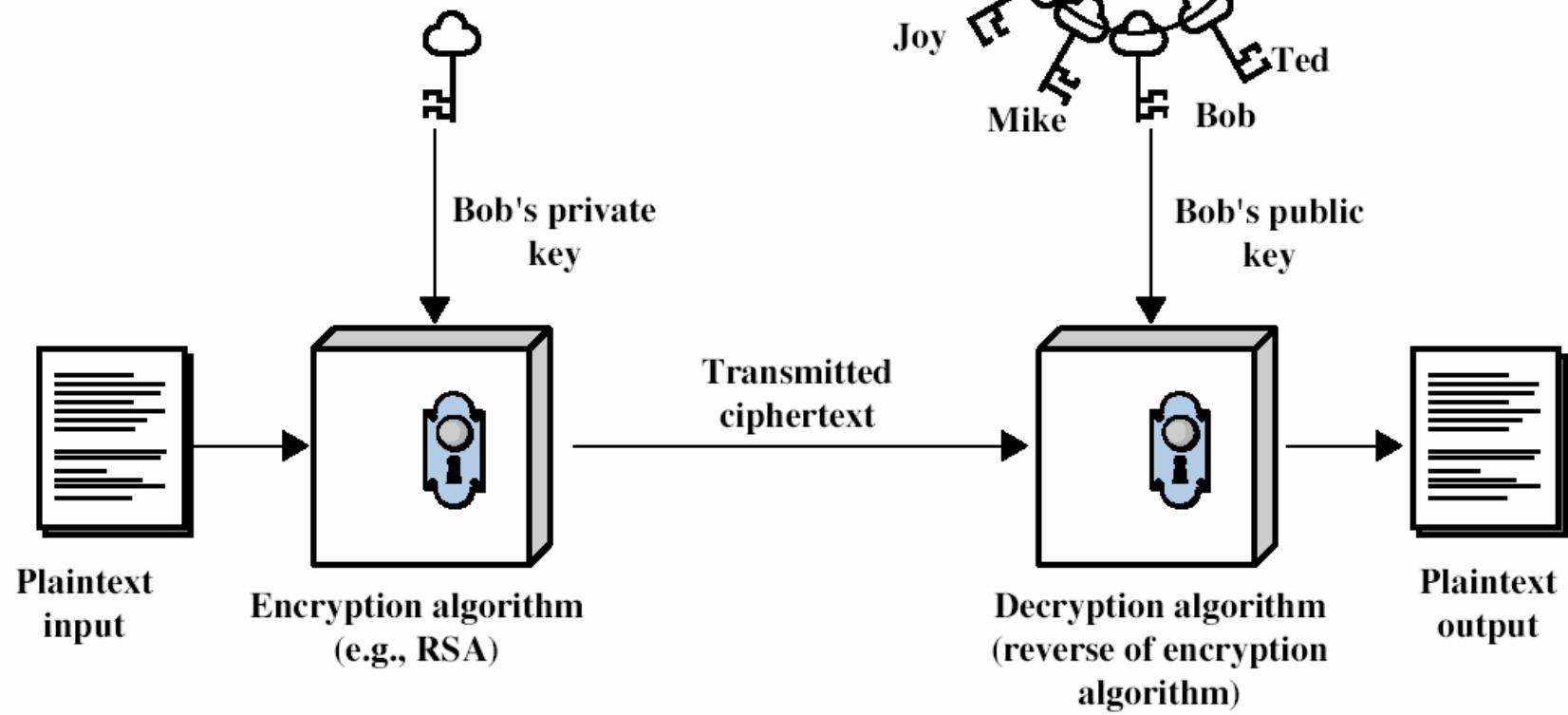
Crittografia a chiave pubblica

Crittografia



Crittografia a chiave pubblica

Autenticazione



Perchè è stata inventata la crittografia a due chiavi??

- Per risolvere due problemi fondamentali:
 - **La distribuzione della chiave** – come possono avversi comunicazioni sicure senza dover porre fiducia in un KDC?
 - **Firme digitali** – come è possibile verificare che un messaggio proviene in forma intatta dal presunto mittente?

Crittografia a chiave pubblica

- Fu introdotta nel 1976 da Whitfield Diffie e Martin Hellman, all'epoca ricercatori presso la Stanford University (California)



Crittografia a chiave pubblica

- Di fatto, il concetto di crittografia a chiave pubblica era già noto alla NSA dalla metà degli anni 60.
- Il lavoro di Diffie&Hellman indusse gli studiosi di crittografia a progettare schemi a chiave pubblica.
- Nel 1977, Ron **Rivest**, Adi **Shamir** e Len **Adleman** del MIT concepirono l'algoritmo RSA

Caratteristiche

- Gli algoritmi a chiave pubblica si basano su due chiavi con le seguenti caratteristiche
 - È computazionalmente irrealizzabile trovare la chiave di decriptazione sulla base della conoscenza dell'algoritmo e della chiave di criptazione
 - È computazionalmente facile criptare e decriptare i messaggi se si conoscono le rispettive chiavi
 - Una delle due chiavi può essere usata per la cifratura, e l'altra deve essere usata per la decifratura (solo però in alcuni schemi)

Combinazione di segretezza e autenticazione

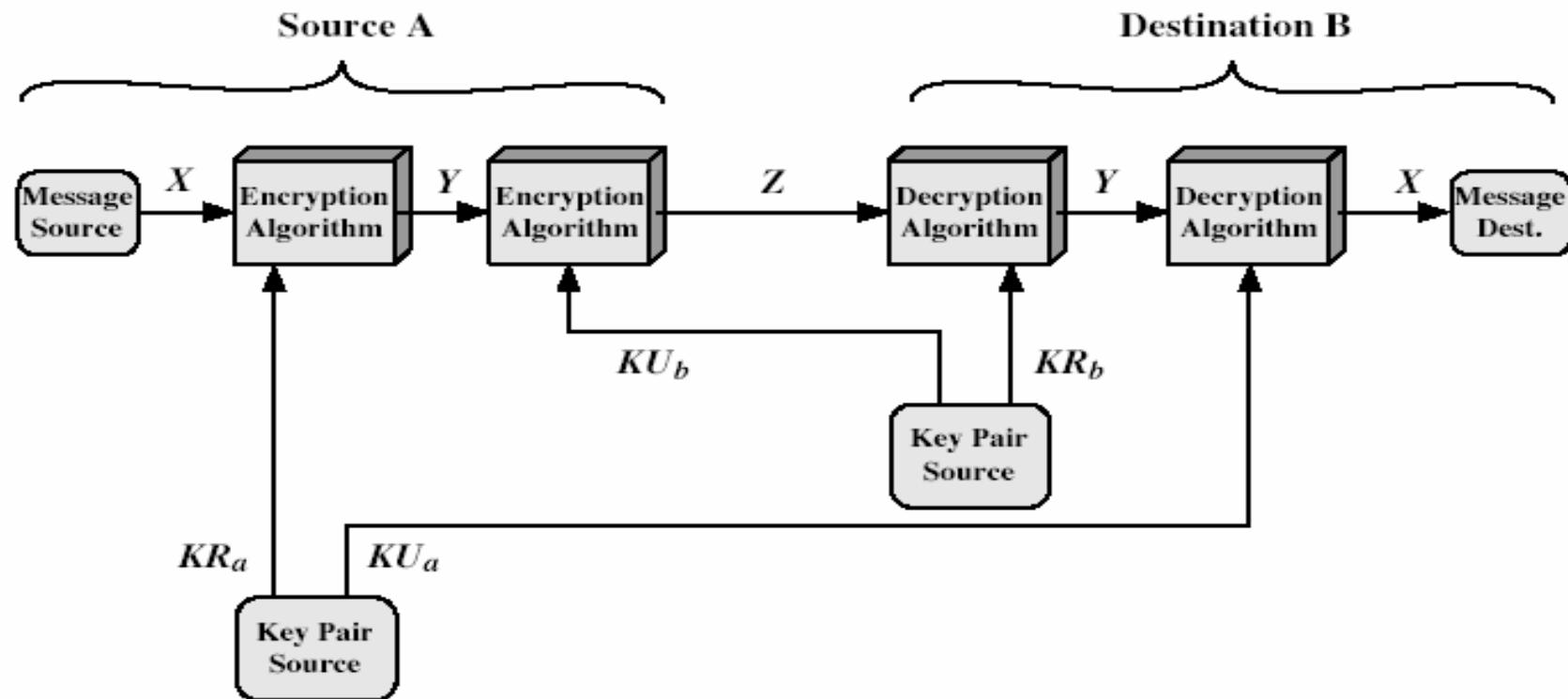


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Applicazioni della Crittografia a chiave pubblica

- Gli usi rientrano in tre categorie:
 - **Criptazione/decriptazione** (per ottenere la segretezza)
 - **Firma digitale** (per ottenere l'autenticazione)
 - **Scambio di chiavi** (ovvero, di chiavi di sessione)
- Alcuni algoritmi possono essere usati per tutti e tre gli usi, altri no.

Sicurezza degli schemi a chiave pubblica

- Come nella crittografia simmetrica, l'**attacco a forza bruta** è sempre teoricamente possibile
- Si usano chiavi molto lunghe (>512bits)
- La sicurezza si basa sulla difficoltà della criptoanalisi
- In generale, si sa come rompere il codice, solo che ciò è talmente complicato da essere irrealizzabile
- Bisogna utilizzare **numeri molto grandi**
- Gli schemi a chiave pubblica sono **più lenti** degli schemi a chiave segreta

RSA

- by Rivest, Shamir & Adleman del MIT in 1977
- È lo schema a chiave pubblica maggiormente noto ed usato
- È basato sull'elevazione a potenza in aritmetica modulo un intero molto grande
 - N.B. l'esponenziazione richiede $O((\log n)^3)$ operazioni (è facile)
- Utilizza interi dell'ordine di 1024 bit
- La sicurezza è basata sulla difficoltà nel fattorizzare grandi numeri interi
 - N.B. la fattorizzazione richiede $O(e^{\log n \log \log n})$ operazioni (è complicata)

RSA

- Il testo in chiaro è cifrato a blocchi; ogni blocco è un intero minore di un dato n
- Ovvero il blocco è lungo k bit con $2^k < n$
- Detto M il plaintext e C il ciphertext, le operazioni di crittografia e decrittografia si esprimono:

$$C = M^e \bmod n$$

$$M = C^d \bmod n = M^{ed} \bmod n$$

RSA

- n è noto sia al mittente che al destinatario
- Il mittente conosce e (che è pubblica)
- **Solo** il destinatario conosce d
- Quindi $KR=\{d, n\}$, $KU=\{e, n\}$

RSA

- Requisiti:
 - È possibile trovare i valori di n , d , e tali che $M^{ed} \bmod n = M$,
 - È relativamente facile calcolare M^e e C^d per tutti i valori di $M < n$
 - È computazionalmente impossibile determinare d sulla base di e ed n

Scelta delle chiavi in RSA (1/2)

- Le due chiavi di ciascun utente sono scelte come segue:
- Si generano due numeri primi molto grandi: p, q
- Si pone poi $n=pq$
 - Nota che $\Phi(n) = (p-1)(q-1)$
- Si seleziona in maniera aleatoria la chiave e
 - ove $1 < e < \Phi(n)$, $\gcd(e, \Phi(n)) = 1$

Scelta delle chiavi in RSA (2/2)

- La chiave di decriptazione d è l'inversa di e modulo $\Phi(n)$
 - $d = e^{-1} \pmod{\Phi(n)}$ con $0 \leq d \leq n$
- La chiave pubblica di criptazione è:
 $KU = \{e, n\}$
- La chiave segreta di decriptazione è:
 $KR = \{d, p, q\}$

Uso di RSA

- Per criptare un messaggio M il mittente:
 - Si procura la **chiave pubblica** del destinatario
 $KU = \{e, n\}$
 - calcola: $C = M^e \text{ mod } n$, ove $0 \leq M < n$
- Per decriptare il testo cifrato C il destinatario:
 - utilizza la sua chiave privata $KR = \{d, p, q\}$
calcolando la quantità $M = C^d \text{ mod } n$
- N.B. Deve essere $M < n$

RSA: Perchè funziona? (1/2)

- Per il teorema di Eulero è: $a^{\Phi(n)} \text{mod } n = 1$
 - ove è $\gcd(a, n) = 1$
- Un corollario del teorema di Eulero, che non dimostriamo, stabilisce che
 - Dati due numeri primi p e q e gli interi $n=pq$, $m < n$, e $k > 0$, vale la relazione

$$m^{k\Phi(n)+1} = m \text{ mod } n$$

RSA: Perchè funziona? (2/2)

- in RSA si ha:
 - $n=pq$
 - $\Phi(n)=(p-1)(q-1)$
 - e e d sono selezionati in modo che siano inversi in aritmetica modulo $\Phi(n)$
 - Quindi è $ed=1+k\Phi(n)$ per qualche k
- Da cui:
$$\begin{aligned} C^d &= (M^e)^d = M^{1+k\Phi(n)} = M \cdot (M^{\Phi(n)})^k = \\ &= M \cdot (1)^k = M^1 = M \bmod n \end{aligned}$$

RSA

- In sintesi, i valori di n ed e sono noti
- d si potrebbe ottenere calcolando l'inverso di e modulo $\Phi(n)$
- Tuttavia $\Phi(n)$ è incalcolabile, a meno che non si sappia che $n = pq$ e che quindi
$$\Phi(n) = (p-1)(q-1)$$
- Ecco perchè la forza di RSA si basa sul fatto che è difficile fattorizzare grandi numeri interi.

RSA: Un esempio

1. Seleziona i numeri primi: $p=17$ & $q=11$
2. Calcola $n = pq = 17 \times 11 = 187$
3. Calcola $\Phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Seleziona e : $\gcd(e, 160) = 1$; scegli $e=7$
5. Determina d : $de \equiv 1 \pmod{160}$ e $d < 160$ tale valore è $d=23$ poichè $23 \times 7 = 161 = 10 \times 160 + 1$
6. La chiave pubblica è $KU = \{7, 187\}$
7. La chiave privata è $KR = \{23, 17, 11\}$

RSA: Un Esempio

- Esempio di criptazione e decriptazione
- Si consideri il messaggio $M = 88$ (n.b. $88 < 187$)
- criptazione:

$$C = 88^7 \bmod 187 = 11$$

- decriptazione:

$$M = 11^{23} \bmod 187 = 88$$

Esponenziazione modulare

- Algoritmo “Square and Multiply”
- Permette di effettuare l'esponenziazione in modo veloce ed efficiente
- La base della potenza viene ripetutamente elevata al quadrato
- Vengono poi moltiplicati insieme solo i valori necessari alla formazione del risultato finale
- Si utilizza la rappresentazione binaria dell'esponente
- La complessità è $O(\log_2 n)$
 - eg. $7^5 = 7^4 \cdot 7^1 = 3 \cdot 7 = 10 \bmod 11$
 - eg. $3^{129} = 3^{128} \cdot 3^1 = 5 \cdot 3 = 4 \bmod 11$

RSA: Generazione delle chiavi

- Per usare RSA si deve:
 - Determinare in modo aleatorio 2 numeri primi p e q
 - selezionare e o d e calcolare l'altro
- I primi p, q non devono essere facilmente derivabili dal prodotto $n=p \cdot q$
 - Devono quindi essere sufficientemente elevati
 - Si usa il test di primalità di Miller-Rabin
- Tipicamente e si sceglie piccolo, in modo che d sia grande

Sicurezza di RSA

- Vi sono 3 strategie di attacco a RSA:
 - Ricerca a forza bruta (non è fattibile perché la chiave è lunga)
 - Attacchi matematici (cercano di fattorizzare n , o di calcolare $\Phi(n)$)
 - Attacchi a tempo (carpiscono informazioni dai tempi impiegati per la decriptazione)

Attacco matematico

- L'attacco matematico si svolge in 3 modi:
 - Si fattorizza $n=pq$, e quindi si trova $\Phi(n)$ e poi d
 - Si trova $\Phi(n)$ direttamente e quindi d
 - Si cerca direttamente d
- Si ritiene che la difficoltà di tali attacchi sia la stessa
- RSA è sicuro se n è lungo circa 1000 bit

Attacchi a tempo

- Si sono sviluppati dalla metà degli anni '90
- Misura i tempi di esecuzione delle esponenziazioni
- Possibili contromisure:
 - Utilizzo di un tempo di esponenziazione costante
 - Introduzione di ritardi aleatori
 - Inserimento di valori spuri nei calcoli

Gestione della chiave

- La crittografia a chiave pubblica aiuta a risolvere il problema della distribuzione delle chiavi
- Dobbiamo occuparci...
 - Della distribuzione delle chiavi pubbliche
 - Dell'uso della crittografia a chiave pubblica per distribuire chiavi di sessione

Distribuzione delle chiavi pubbliche

- Vi sono 4 possibili strategie
 - Annuncio pubblico
 - Elenco pubblico
 - Autorità di distribuzione delle chiavi pubbliche
 - Certificati con chiave pubblica

Annuncio pubblico

- Gli utenti distribuiscono le chiavi pubbliche tramite messaggi di broadcast o in allegato ai loro messaggi
 - e.g., aggiungendo le chiavi alla fine delle loro e-mail o sui newsgroup
- Vi è però un grande punto debole....
 - Chiunque può spacciarsi per un altro e pubblicare una chiave falsa
 - Un utente malizioso può quindi spacciarsi per un altro

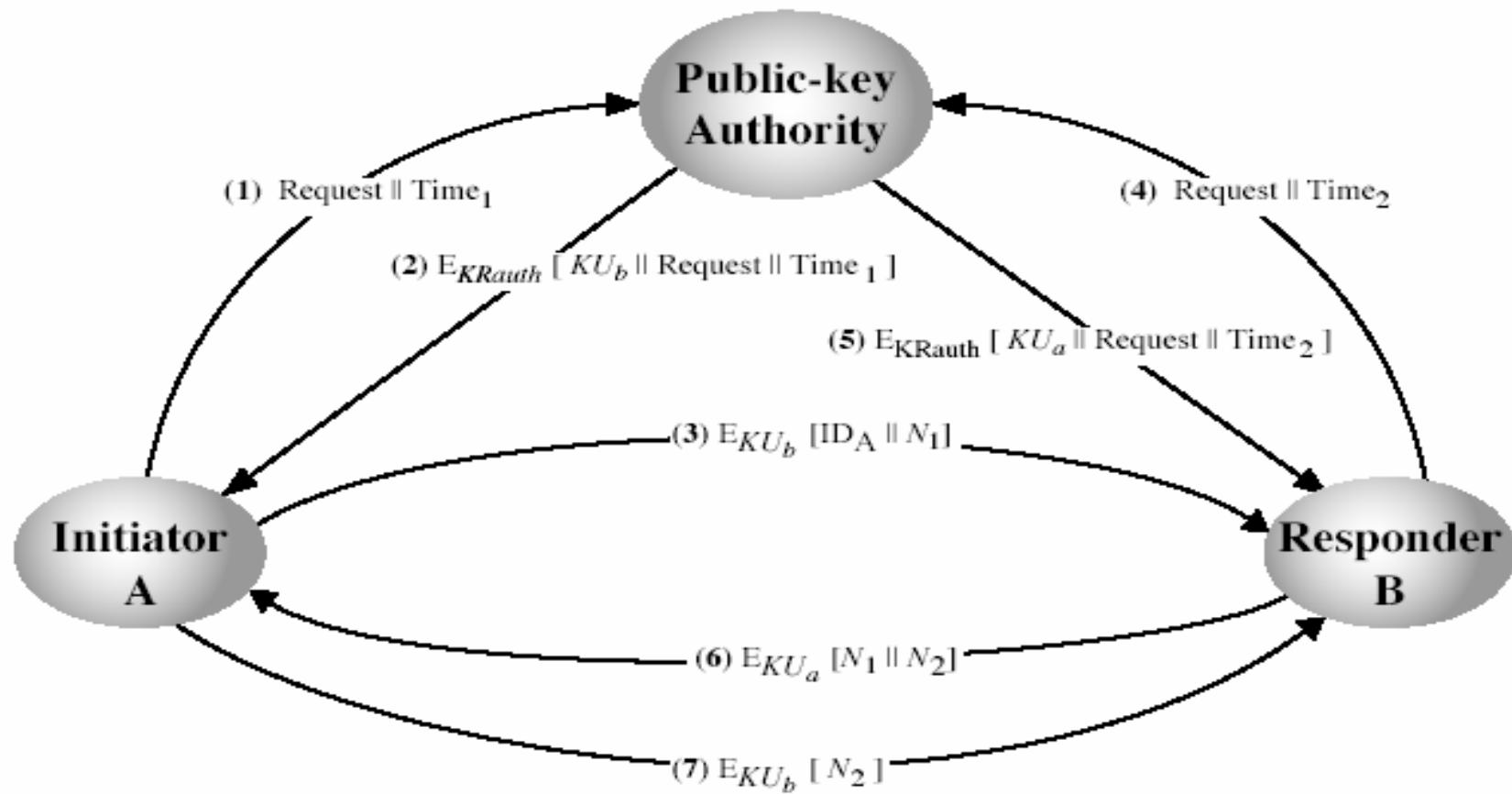
Elenco Pubblico

- Maggiore sicurezza si ha registrando le chiavi presso un elenco pubblico
- Tale elenco è tale che:
 - Contiene voci del tipo {nome,chiave pubblica}
 - Gli utenti registrano la loro chiave in forma sicura
 - Gli utenti possono sostituire la chiave quando vogliono
 - L'elenco viene pubblicato periodicamente
 - L'elenco può essere consultato elettronicamente (in modo protetto)
- L'elenco pubblico può comunque essere violato

Autorità di distribuzione delle chiavi pubbliche

- Migliora la sicurezza mediante un controllo più rigido sulla distribuzione delle chiavi
- Si tratta ancora di un database di chiavi
- Gli utenti devono conoscere la chiave pubblica dell'autorità di distribuzione
- Gli utenti interagiscono con l'autorità per ottenere in modo sicuro la chiave pubblica di altri utenti
 - È richiesto quindi l'accesso in tempo reale al database ogni volta che sono richieste delle chiavi

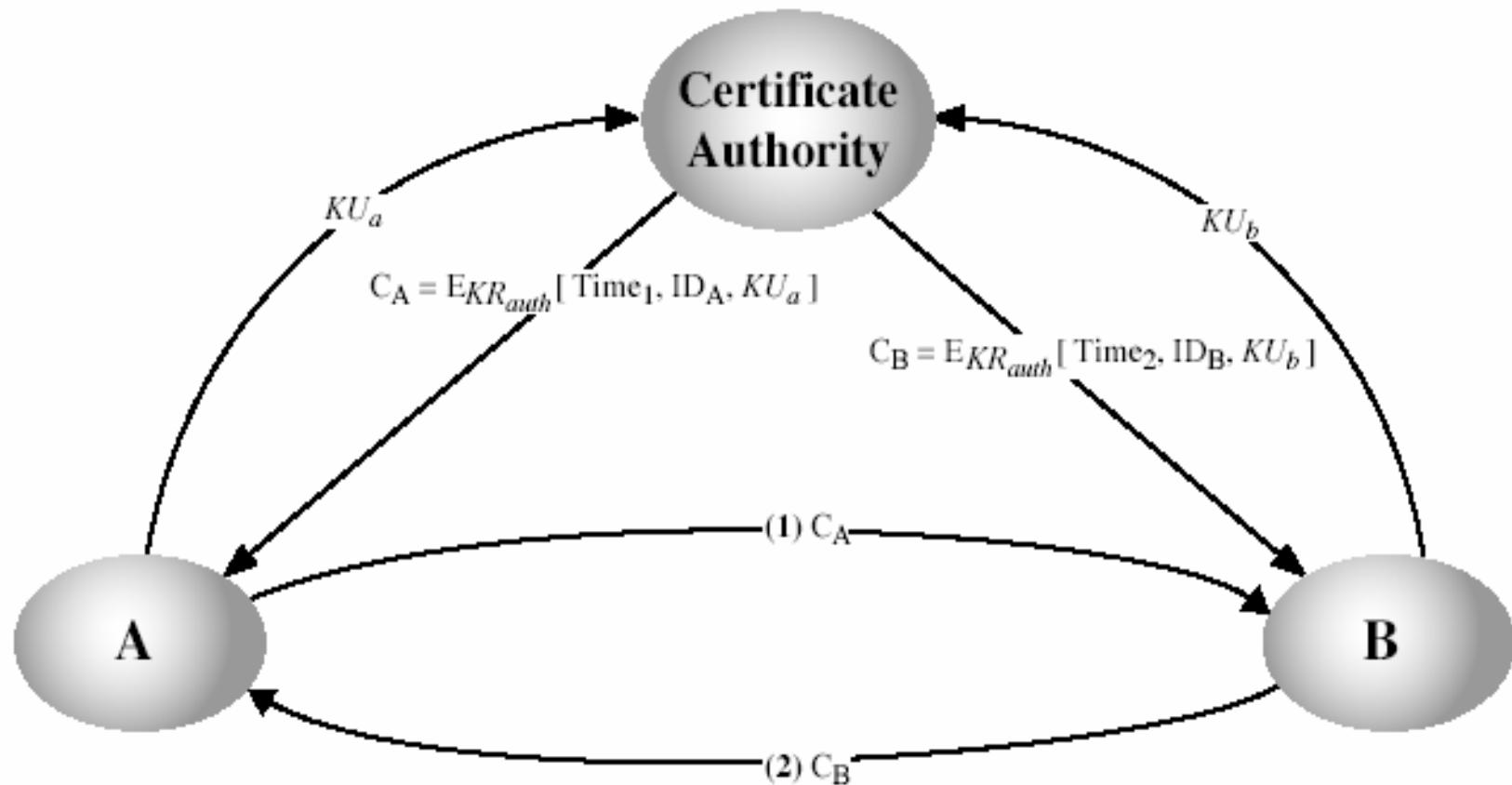
Autorità di distribuzione delle chiavi pubbliche



Certificati a chiave pubblica

- Permettono lo scambio delle chiavi senza bisogno di dover accedere in tempo reale al database pubblico appena considerato
- I certificati possono essere usati direttamente dagli utenti per scambiarsi le chiavi
- Ciascun certificato contiene una chiave pubblica, informazioni supplementari (periodo di validità, condizioni di uso, etc.), e la firma dell'**autorità di certificazione (CA)**
- Un utente invia agli altri la propria chiave trasmettendo il proprio certificato

Certificati a chiave pubblica



Distribuzione delle chiavi segrete tramite chiavi pubbliche

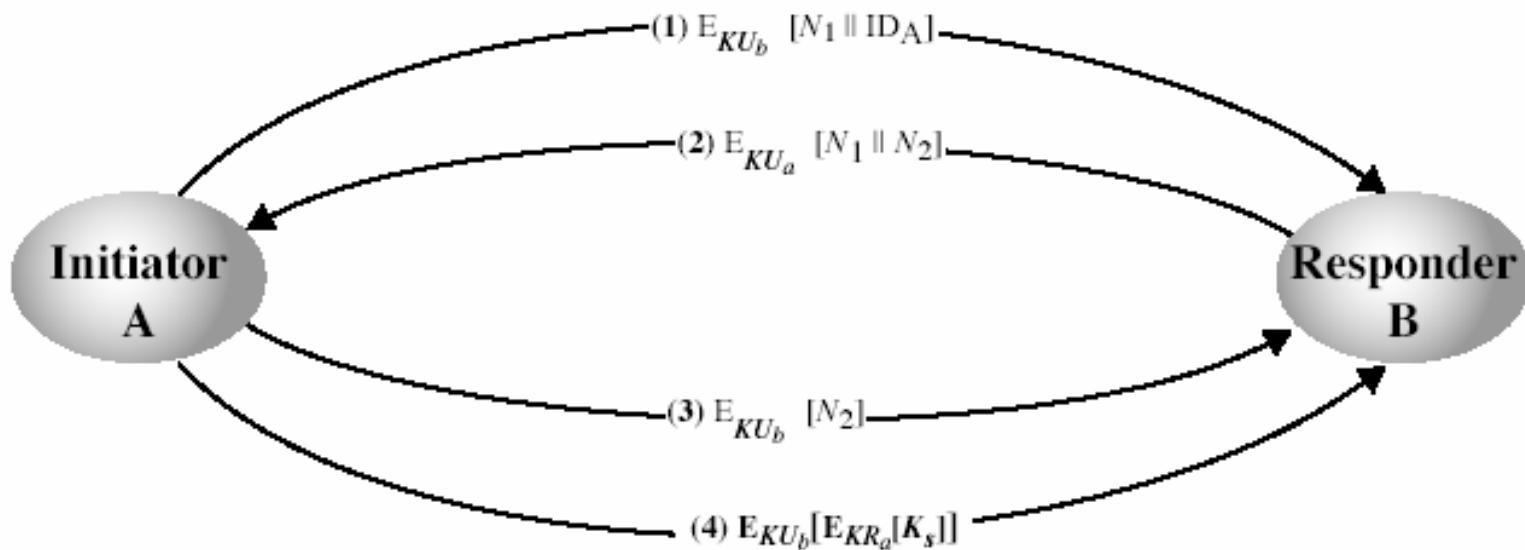
- I metodi presentati permettono di ottenere la chiave pubblica
- Possono essere utilizzati per la segretezza e l'autenticazione
- ...ma gli algoritmi a chiave pubblica sono lenti
- Quindi si preferisce commutare sulla crittografia a chiave segreta
- Bisogna quindi scambiarsi una chiave di sessione
- Vi sono varie strategie...

Semplice distribuzione (Merkle, 1979)

- proposta da Merkle nel 1979
 - A genera una coppia di chiavi
 - A invia a B la chiave pubblica e la sua identità (*in forma non protetta!!!*)
 - B genera una chiave di sessione K e la invia ad A criptata con la chiave pubblica di A
 - A decripta la chiave di sessione e comincia la comunicazione
- Il problema è che un intruso può intercettare il primo messaggio e impersonare entrambi i soggetti della comunicazione

Distibuzione della chiave con segretezza e autenticazione

- Si utilizza in tal caso la crittografia a chiave pubblica



Scambio delle chiavi Diffie-Hellman

- È il primo tipo di schema a chiave pubblica
- by Diffie & Hellman in 1976 insieme all'esposizione del concetto di crittografia a chiave pubblica
 - N.B.: dal 1987 si sa che James Ellis (UK CESG) aveva proposto tale tecnica nel 1970
- È un metodo pratico per lo scambio pubblico di una chiave segreta
- Usato in molti prodotti commerciali

Scambio delle chiavi Diffie-Hellman

- Lo schema di Diffie-Hellman
 - Non può essere usato per scambiarsi un messaggio arbitrario
 - Ma solo per stabilire una chiave nota solo ai due soggetti della comunicazione
- È basato sul concetto di esponenziazione in un campo finito di Galois
- La sua sicurezza si basa sulla difficoltà nel calcolare i logaritmi discreti

Diffie-Hellman: Inizializzazione

- Gli utenti concordano dei parametri comuni e noti:
 - Un grande numero primo o polinomio q
 - α , una radice primitiva di q
- Ogni utente (eg. A) genera la sua chiave
 - sceglie un numero: $x_A < q$
 - Calcola la **chiave pubblica**: $y_A = \alpha^{x_A} \text{ mod } q$
- A rende poi nota la chiave y_A

Diffie-Hellman: Scambio delle chiavi

- La chiave di sessione per A e B è K_{AB} :
$$\begin{aligned} K_{AB} &= \alpha^{x_A \cdot x_B} \bmod q \\ &= y_A^{x_B} \bmod q \quad (\text{che B può calcolare}) \\ &= y_B^{x_A} \bmod q \quad (\text{che A può calcolare}) \end{aligned}$$
- K_{AB} è poi usata come chiave di sessione in uno schema di crittografia a chiave segreta tra Alice e Bob
- L'attacco a tale schema richiede che venga calcolato x_A , ovvero il logaritmo discreto di y_A in base α e modulo q

Diffie-Hellman: Esempio

- Alice & Bob vogliono scambiarsi le chiavi;
- Concordano su $q=353$ e $\alpha=3$
- Selezionano in modo random le chiavi segrete:
 - A sceglie $x_A=97$, B sceglie $x_B=233$
- Calcolano le chiavi pubbliche:
 - $y_A = 3^{97} \text{ mod } 353 = 40$ (Alice)
 - $y_B = 3^{233} \text{ mod } 353 = 248$ (Bob)
- Calcolano la chiave di sessione segreta:
$$K_{AB} = y_B^{x_A} \text{ mod } 353 = 248^{97} \text{ mod } 353 = 160 \quad (\text{Alice})$$
$$K_{AB} = y_A^{x_B} \text{ mod } 353 = 40^{233} \text{ mod } 353 = 160 \quad (\text{Bob})$$

Autenticazione dei messaggi e funzioni hash

Autenticazione

- L'autenticazione serve a:
 - Proteggere l'integrità del messaggio
 - Validare l'identità del mittente
 - Garantire la non ripudiabilità
- In generale, tre possibili approcci sono possibili:
 - Criptazione dei messaggi
 - message authentication code (MAC)
 - Funzioni hash

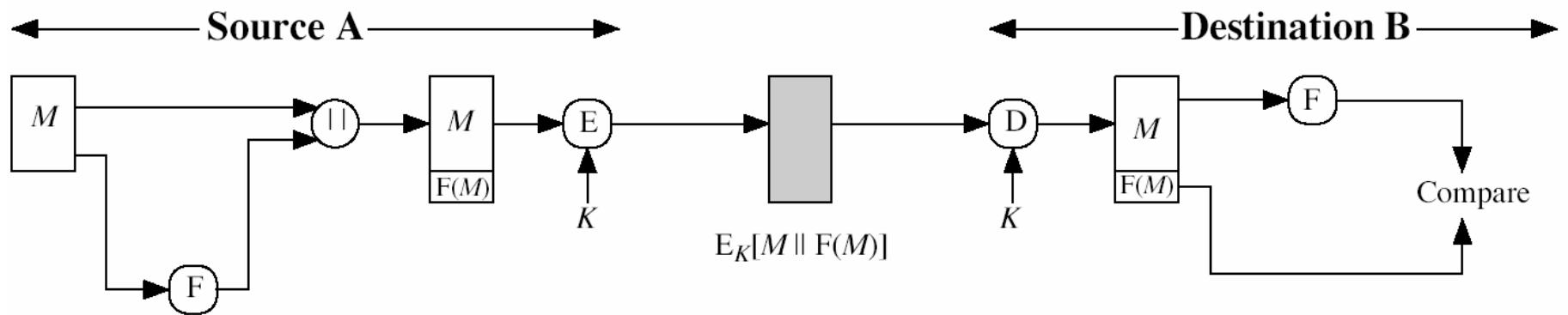
Requisiti di Sicurezza

- In generale, sono possibili i seguenti attacchi
 - Violazione della segretezza
 - Analisi del traffico
 - Mascheramento (inserzione di messaggi fasulli)
 - Modifica dei contenuti
 - Modifica della sequenza dei messaggi
 - Modifica temporale
 - Ripudiazione dell'origine (l'origine nega di aver trasmesso il messaggio)
 - Ripudiazione della destinazione (la destinazione nega di aver trasmesso il messaggio)

Criptazione dei messaggi

- Abbiamo già visto che la criptazione è capace di fornire in una certa misura anche l'autenticazione
- Se è utilizzata la crittografia simmetrica:
 - Il destinatario sa che l'informazione arriva dal mittente
 - Se il messaggio è strutturato (ad esempio mediante bit di ridondanza) è possibile rivelarne eventuali alterazioni

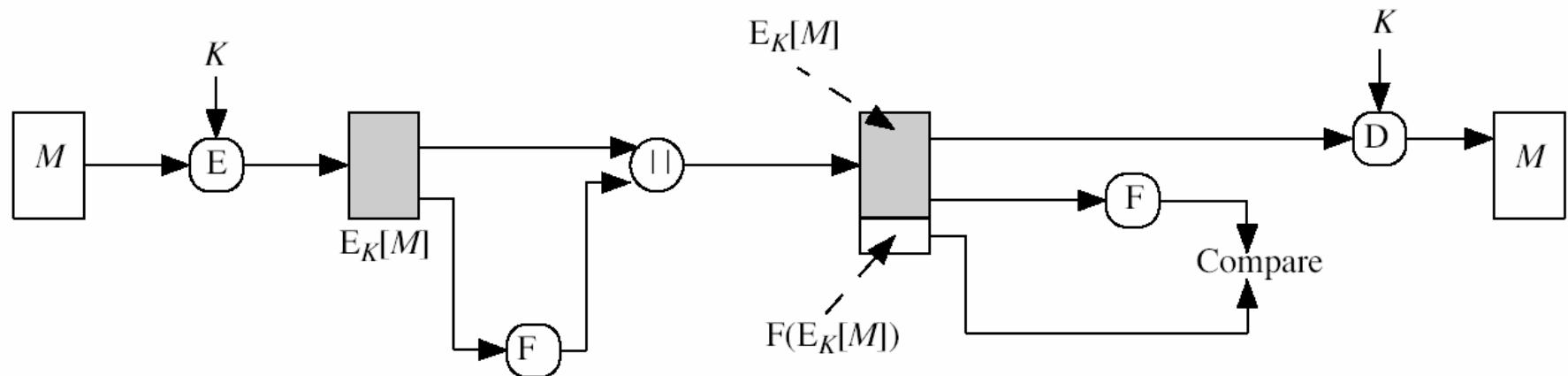
Controllo degli errori interno



(a) Internal error control

In questo caso si garantisce l'autenticazione

Controllo degli errori esterno



(b) External error control

Criptazione dei messaggi

- Se è utilizzata la crittografia a due chiavi:
 - La criptazione con la chiave privata del mittente non assicura la segretezza
 - Tuttavia se:
 - Il mittente **firma** il messaggio con la proprie chiave privata
 - Poi lo **cripta** con la chiave pubblica del destinatario
 - Si consegue sia la segretezza sia l'autenticazione

Criptazione dei messaggi

- Se il ricevente conserva una copia del messaggio criptato, può dimostrare che è stato inviato dal mittente!!
- C'è ancora bisogno di riconoscere eventuali messaggi alterati
- Tale schema richiede un totale di 2 crittografie e decrittografie per messaggio inviato

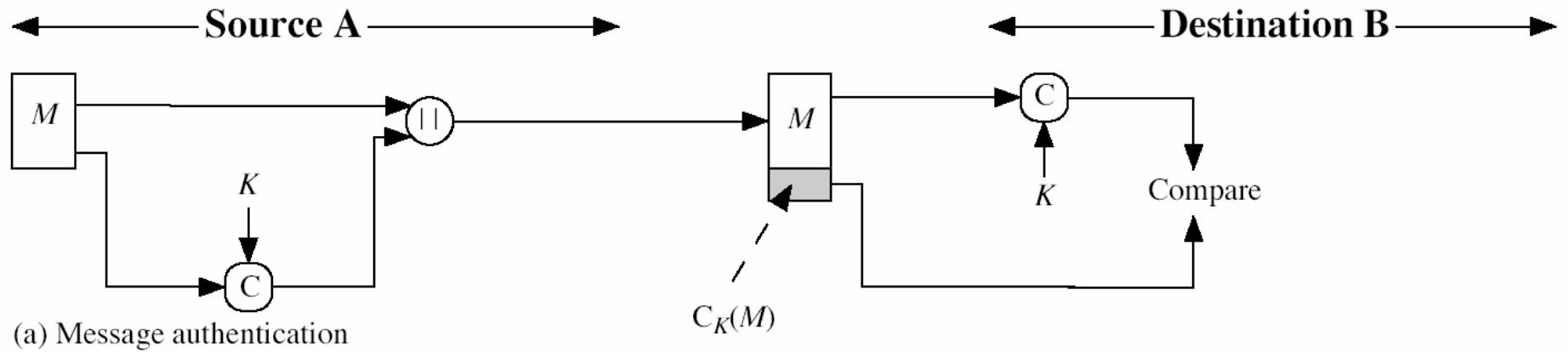
Message Authentication Code (MAC)

- E' una tecnica alternativa: viene generato un piccolo blocco di dati che
 - Dipende sia dal messaggio che da una chiave segreta
 - Differisce dalla criptazione perchè tale trasformazione non deve essere reversibile
- Il MAC viene aggiunto al messaggio come se fosse una **firma**
- Il destinatario può calcolare il MAC e verificare che il messaggio è giunto integro e proviene dal mittente

Message Authentication Code (MAC)

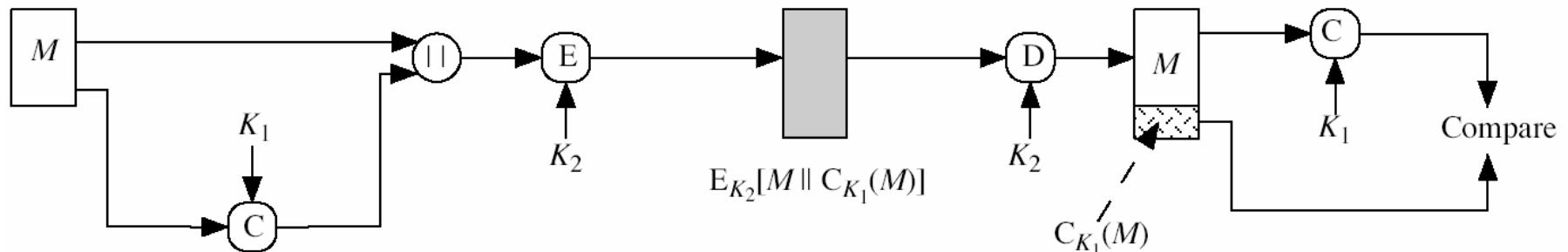
- La funzione MAC è del tipo many-to-one
- Ad esempio, se
 - n è la lunghezza del MAC
 - N è il numero di possibili messaggi
 - k è la lunghezza della chiave
- Allora:
 - Vi saranno solo $2^n << N$ diversi codici MAC
 - Ciascun codice MAC sarà generato mediamente da $N / 2^n$ diversi messaggi
 - Vi saranno 2^k diversi mappaggi dal set dei messaggi al set dei codici MAC

Modi di Uso del MAC



Autenticazione senza segretezza

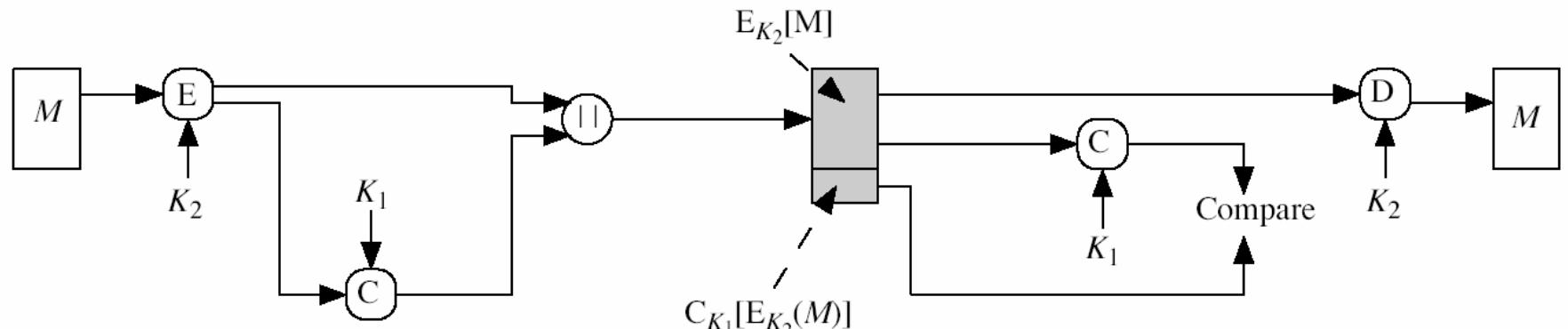
Modi di Uso del MAC



(b) Message authentication and confidentiality; authentication tied to plaintext

Autenticazione e segretezza;
Codice MAC concatenato al plaintext

Modi di Uso del MAC



(c) Message authentication and confidentiality; authentication tied to ciphertext

Autenticazione e segretezza;
Codice MAC concatenato al ciphertext

Codici MAC

- Il MAC può essere quindi abbinato alla crittografia per fornire segretezza
 - In genere si usano chiavi diverse per autenticazione e segretezza
 - In genere si preferisce concatenare il MAC al testo in chiaro
- Perchè usare il MAC per garantire l'autenticazione quando basta anche la crittografia?
 - A volte è richiesta solo l'autenticazione
 - Il MAC richiede minori risorse computazionali
 - A volte l'autenticazione deve essere conservata per verificare ogni volta l'integrità dei dati memorizzati

MAC=firma digitale?

- Si noti che il MAC non fornisce il servizio di firma digitale!
- Infatti, mittente e destinatario condividono la stessa chiave

Proprietà dei MAC

- Il MAC è sostanzialmente un checksum crittografico
 - $\text{MAC} = C_K(M)$
 - Condensa un messaggio di lunghezza variabile
 - Utilizzando una data chiave segreta K
 - Il MAC ha lunghezza fissa
- È una funzione many-to-one
 - Molti messaggi possono avere lo stesso MAC

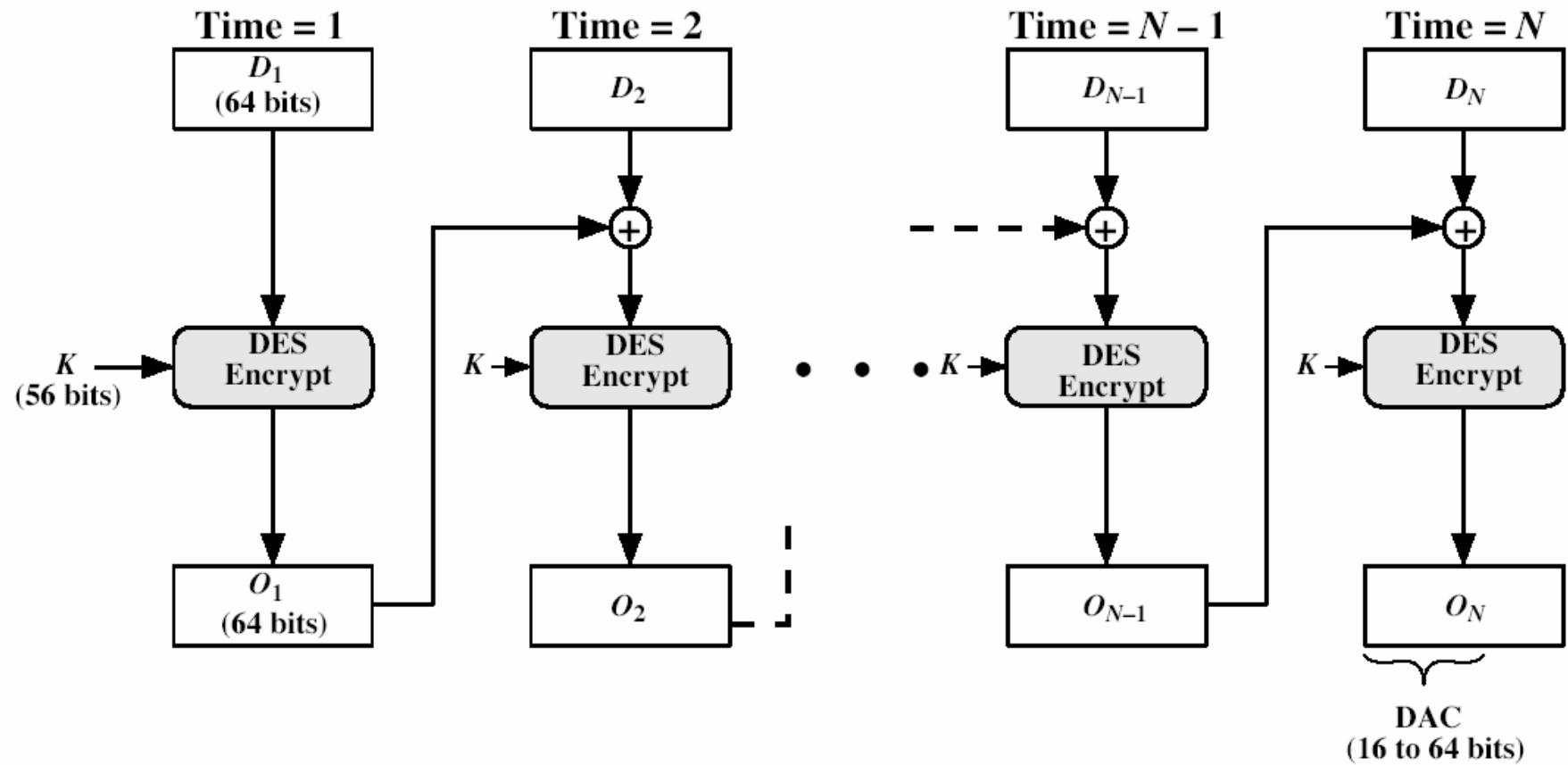
MAC: Esempio di attacco

- Si supponga $k > n$ (la chiave è più lunga del codice MAC)
- Dati M_1 e MAC_1 , l'estraneo calcola il MAC su M_1 per tutte le possibili 2^k chiavi
- Il numero di corrispondenze è 2^{k-n}
- Dati M_2 e MAC_2 , l'estraneo calcola il MAC su M_2 per tutte le 2^{k-n} chiavi candidate
- E così via.....
- N.B. C'è bisogno di diverse coppie di messaggi e dei relativi codici MAC

Requisiti di sicurezza dei MAC

1. Noto un messaggio ed il MAC è impossibile trovare un altro messaggio cui corrisponde lo stesso MAC (assumendo ovviamente che la chiave non sia nota)
2. I MAC devono essere uniformemente distribuiti (ovvero, scelti a caso 2 messaggi, i loro MAC coincidono con probabilità 2^{-n})
3. Il MAC deve dipendere in “egual maniera” da tutti i bit del messaggio

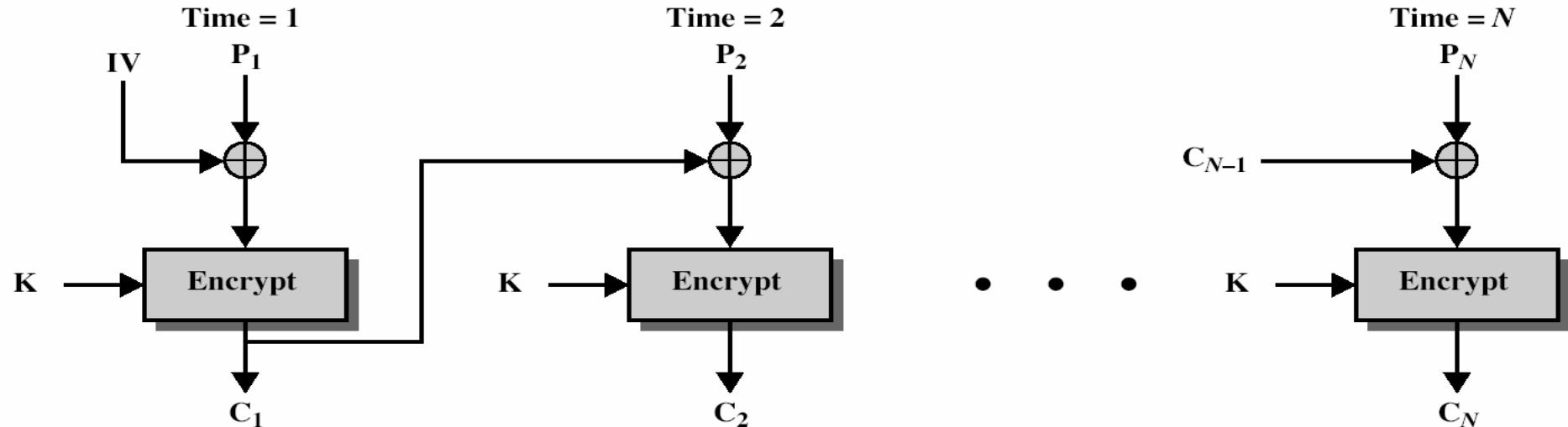
Codici MAC basati su DES



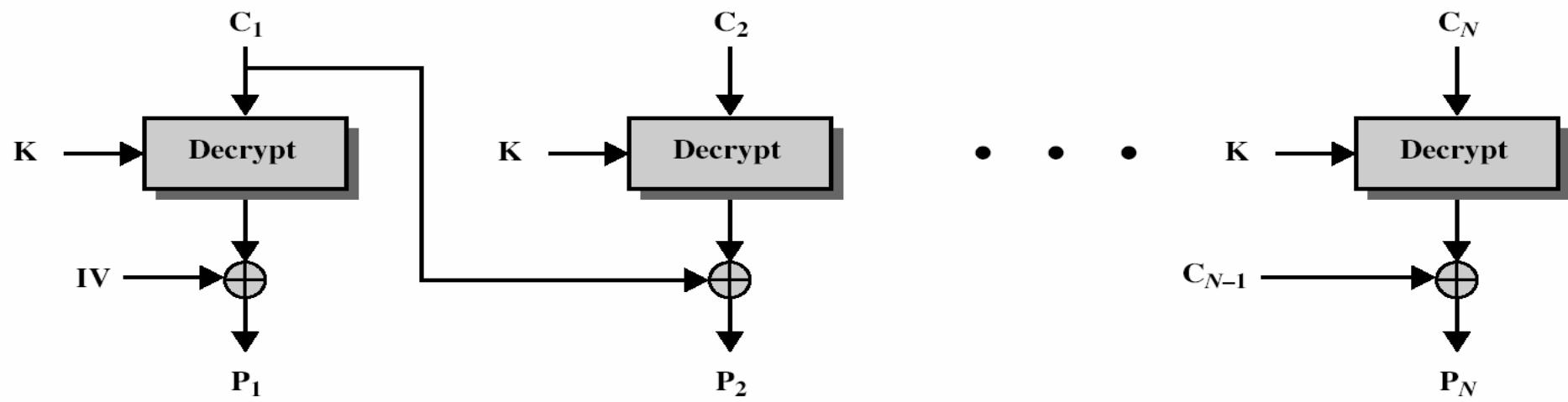
Codici MAC basati su DES

- Si può in realtà utilizzare qualsiasi schema crittografico a blocchi
- **Data Authentication Algorithm (DAA)** è un codice MAC ampiamente usato basato su DES-CBC
 - usa $IV=0$ e uno zero-padding del blocco finale
 - Cripta i messaggi usando DES in modalità CBC
 - Il MAC è il blocco finale, o i primi M bit ($16 \leq M \leq 64$) del blocco finale
- Nota che la lunghezza del MAC finale è però troppo corta per garantire la sicurezza

Cipher Block Chaining (CBC)



(a) Encryption

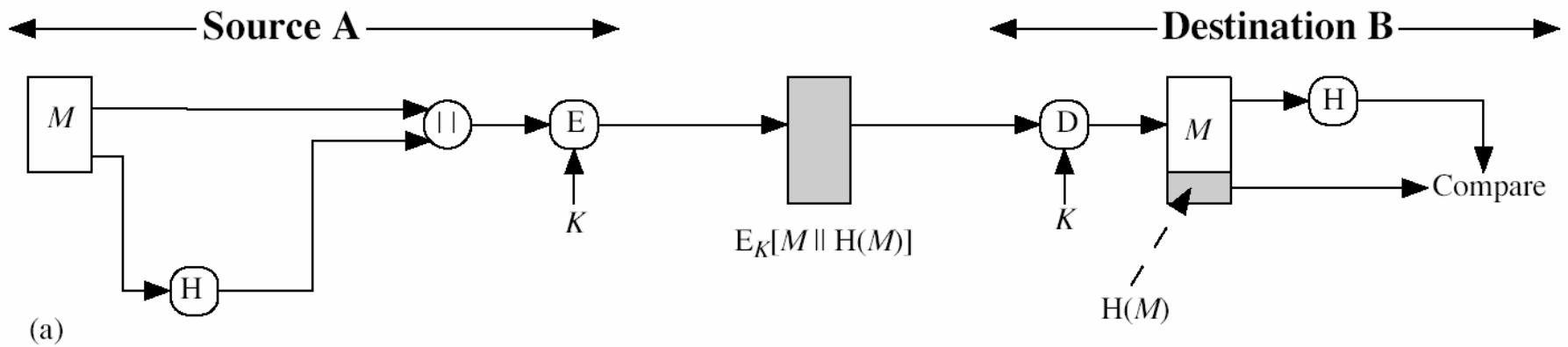


(b) Decryption

Funzione Hash

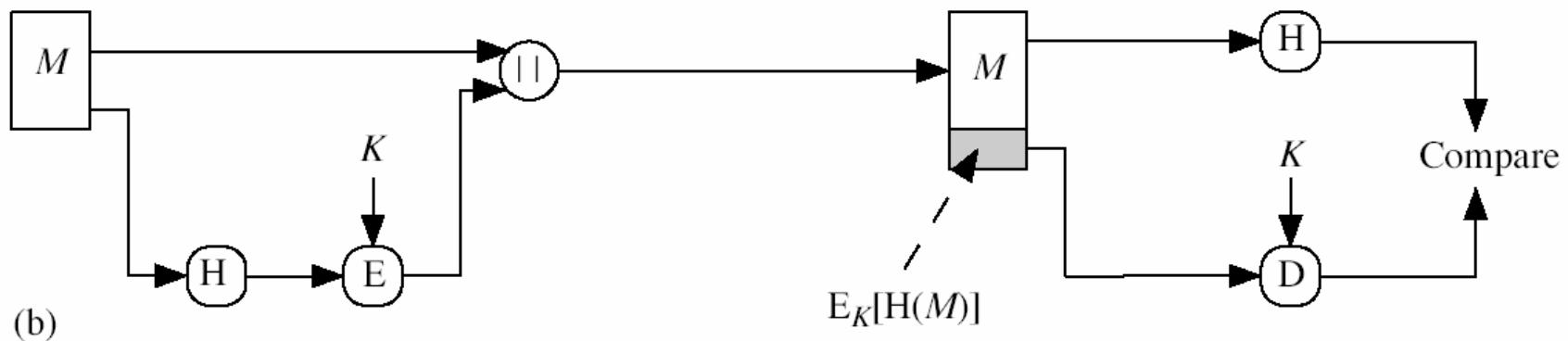
- Associa a messaggi di lunghezza arbitraria una stringa di lunghezza fissa
- A differenza del MAC, la funzione hash è nota e non dipende da alcuna chiave
- Viene usata per rivelare cambi in un messaggio
- Può essere usata in vari modi
- Può servire anche a creare una firma digitale

Uso delle funzioni hash – chiave segreta



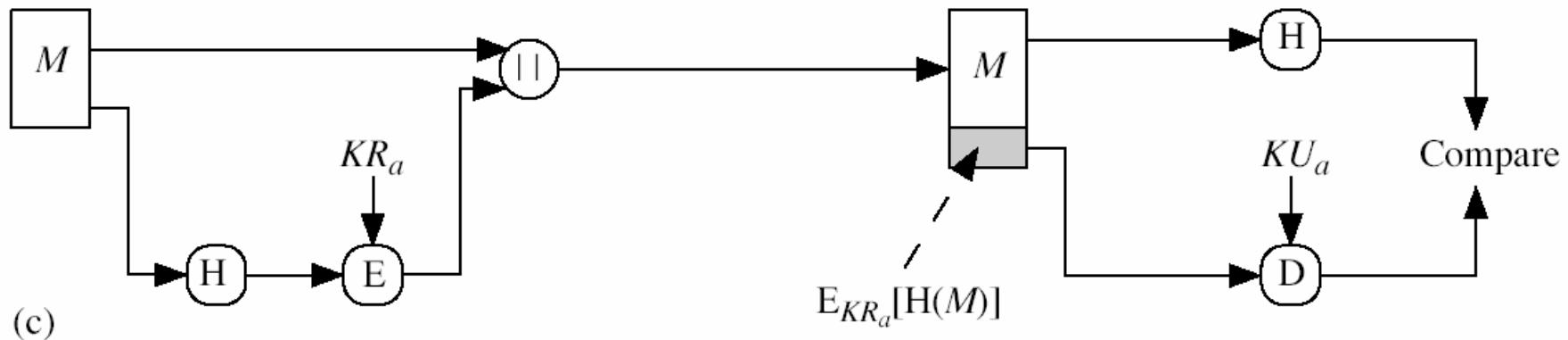
Garantisce segretezza e autenticazione;
B può però coniare un messaggio e dire che è stato
Inviato da A

Uso delle funzioni hash – chiave segreta



Viene crittografato solo il codice hash;
Nota che hash + crittografia = codice MAC

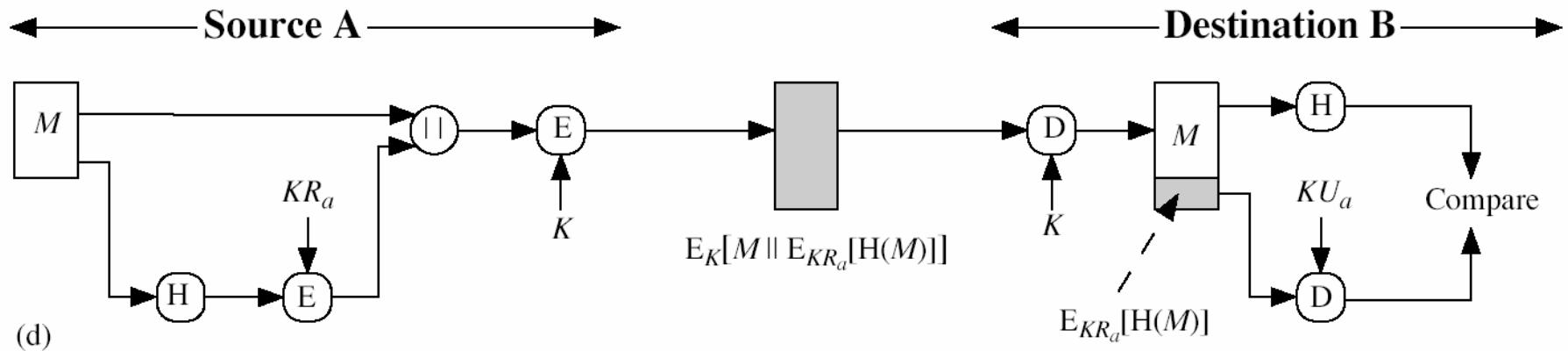
Uso delle funzioni hash – chiave pubblica



Viene crittografata solo la funzione hash con la chiave privata del mittente.

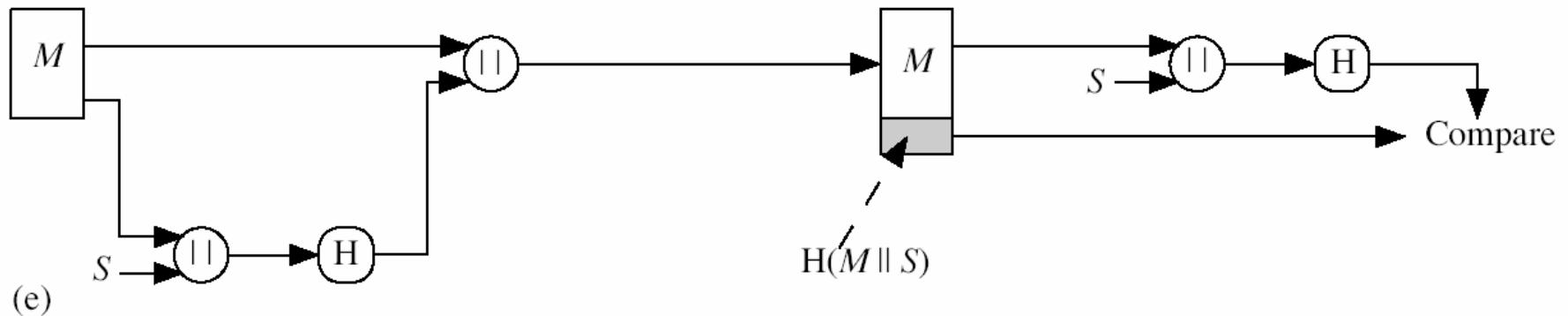
Si tratta della tecnica a firma digitale!!

Uso delle funzioni hash – chiave pubblica + chiave segreta



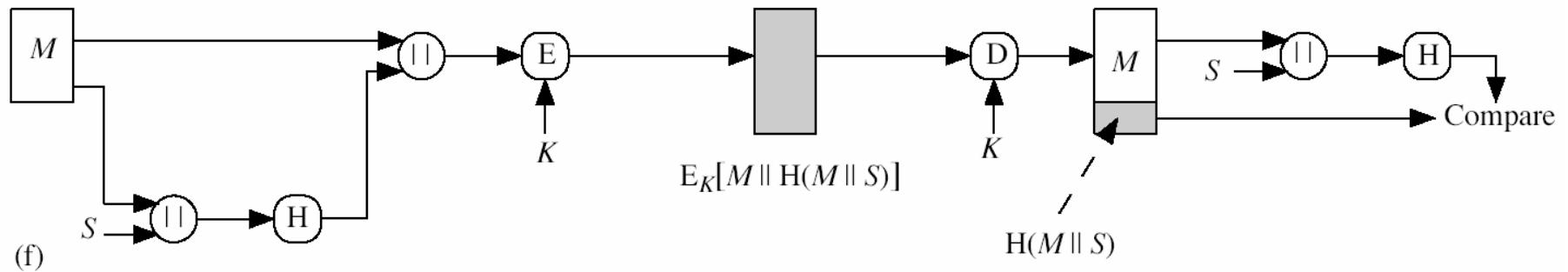
Firma digitale + segretezza con crittografia a chiave segreta
(tale tecnica è ampiamente utilizzata)

Uso delle funzioni hash



Viene utilizzata la funzione hash ma senza la crittografia; si assume che trasmettitore e ricevitore condividano un valore segreto comune S

Uso delle funzioni hash



Tale schema differisce dal precedente per l'aggiunta della segretezza

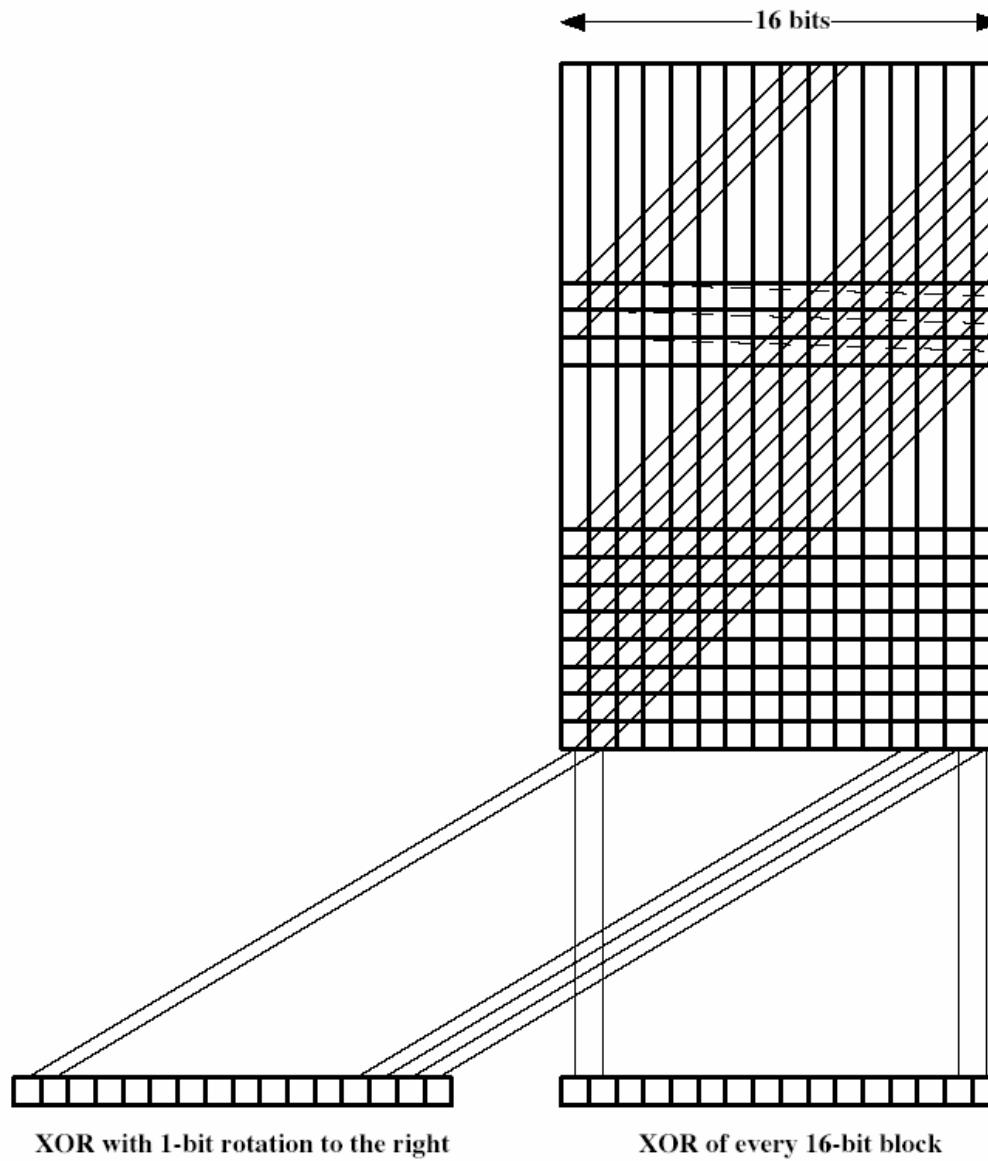
Requisiti delle funzioni Hash

1. Può essere calcolate per messaggi M di qualsiasi dimensione
2. Produce un output h di lunghezza fissa
3. $h = H(M)$ è facile da calcolare qualunque sia M
4. dato h è impossibile trovare $x : H(x) = h$
 - Proprietà “one-way”
5. dato x è impossibile trovare $y : H(y) = H(x)$
 - Resistenza debole alle collisioni
6. È impossibile trovare una coppia $x, y : H(y) = H(x)$
 - Resistenza forte alle collisioni

Semplici funzioni hash

- Una funzione hash molto comune è basata sull'uso della XOR
- Non è una scelta sicura in quanto il messaggio può essere manipolato senza che l'hash cambi
- Tale funzione deve essere quindi accoppiata con una funzione crittografica

Esempio



Attacchi a compleanno

- Le funzioni hash sono vulnerabili all'attacco a compleanno
- L'**attacco a compleanno** opera nel modo seguente:
 - L'origine A firma un messaggio con un codice hash di m bit e crittografandolo con la sua chiave privata
 - L'attaccante genera $2^{m/2}$ varianti del messaggio valido tutte con lo stesso significato
 - L'attaccante genera inoltre $2^{m/2}$ varianti di un messaggio fraudolento che si vuole inviare
 - I due set di messaggi sono confrontati per trovare una coppia con lo stesso hash (la probabilità che ciò accada è > 0.5 per il paradosso del compleanno)
 - L'attaccante può quindi sostituire i messaggi
- La lunghezza del codice hash deve quindi essere notevole

Dear Anthony,

{This letter is} to introduce {you to} {Mr.} Alfred {P.}
{ I am writing } {to you} {--}

Barton, the {new} {chief} jewellery buyer for {our}
{newly appointed} {senior}

Northern {European} {area}. He{will take} over {the}
{Europe} {division}

responsibility for {all} our interests in {watches and jewellery}
{the whole of} {jewellery and watches}

in the {area}. Please {afford} him {every} help he {may need}
{region} {give} {all the} {needs}

to {seek out} the most {modern} lines for the {top} end of the
{find} {up to date} {high}

market. He is {empowered} to receive on our behalf {samples}
{authorized} {specimens} of the

{latest} {watch and jewellery} products, {up} {subject} to a {limit}
{newest} {jewellery and watch}

of ten thousand dollars. He will {carry} a signed copy of this {letter}
{hold} {document}

as proof of identity. An order with his signature, which is {appended}
{attached}

{authorizes} you to charge the cost to this company at the {above}
{allows} {head office}

address. We {fully} expect that our {level} of orders will increase in
{--} {volume}

the {following} year and {trust} that the new appointment will {be}
{next} {hope} {prove}

{advantageous} to both our companies.
{an advantage}

Una
lettera
in 2³⁷
varianti

Il Paradosso del compleanno

- Sia $n=2^m$ il numero di possibili funzioni hash
- Dato un Y , scelto un messaggio X a caso, la probabilità che $H(X)=H(Y)$ è $1/n$; la probabilità che $H(X)\neq H(Y)$ è $(1-1/n)$
- Scelti k valori casuali di X , la probabilità che nessuno di tali messaggi verifichi l'uguaglianza è $(1-1/n)^k$

Il Paradosso del compleanno

- Ne consegue che la probabilità che vi sia almeno una corrispondenza su k è:

$$1-(1-1/n)^k \approx k/n$$

- Ne consegue quindi che, al crescere di k la probabilità di trovare una coincidenza non è trascurabile

Funzioni Hash basate su Cifrari a Blocchi

- I cifrari a blocchi possono essere usati per creare funzioni hash
 - Ad esempio, posto $H_0=0$
 - calcola: $H_i = E_{M_i}[H_{i-1}]$, con M_i blocco i-esimo del messaggio
 - Si usa l'ultimo blocco come codice hash
 - Nota che non c'è una chiave!
- Il codice hash risultante può essere troppo piccolo (64-bit)
 - Vulnerabile ad attacchi crittografici tra cui l'attacco a compleanno

Algoritmi di generazione di funzioni hash

Algoritmi hash

- Vi sono analogie tra l'evoluzione delle funzioni hash e quella dei cifrari simmetrici
 - Aumento dell'efficacia degli attacchi a forza bruta
 - Ne consegue un'evoluzione degli algoritmi
 - e.g., da DES ad AES nei cifrari a blocchi
 - da MD4 & MD5 a SHA-1 & RIPEMD-160 negli algoritmi hash
- Anche gli algoritmi hash usano strutture iterative, analoghe alla struttura di Feistel nei cifrari a blocco

MD5

- Progettato da Ronald Rivest (la R di RSA)
- È successivo ai precursori MD2 e MD4
- Produce un codice hash di 128 bit; l'input è elaborato in blocchi di 512 bit
- È stato uno dei più usati algoritmi hash fino a qualche anno fa
 - Recentemente sono sorti dubbi sulla sua vulnerabilità agli attacchi a forza bruta
- È anche un Internet standard (RFC1321)

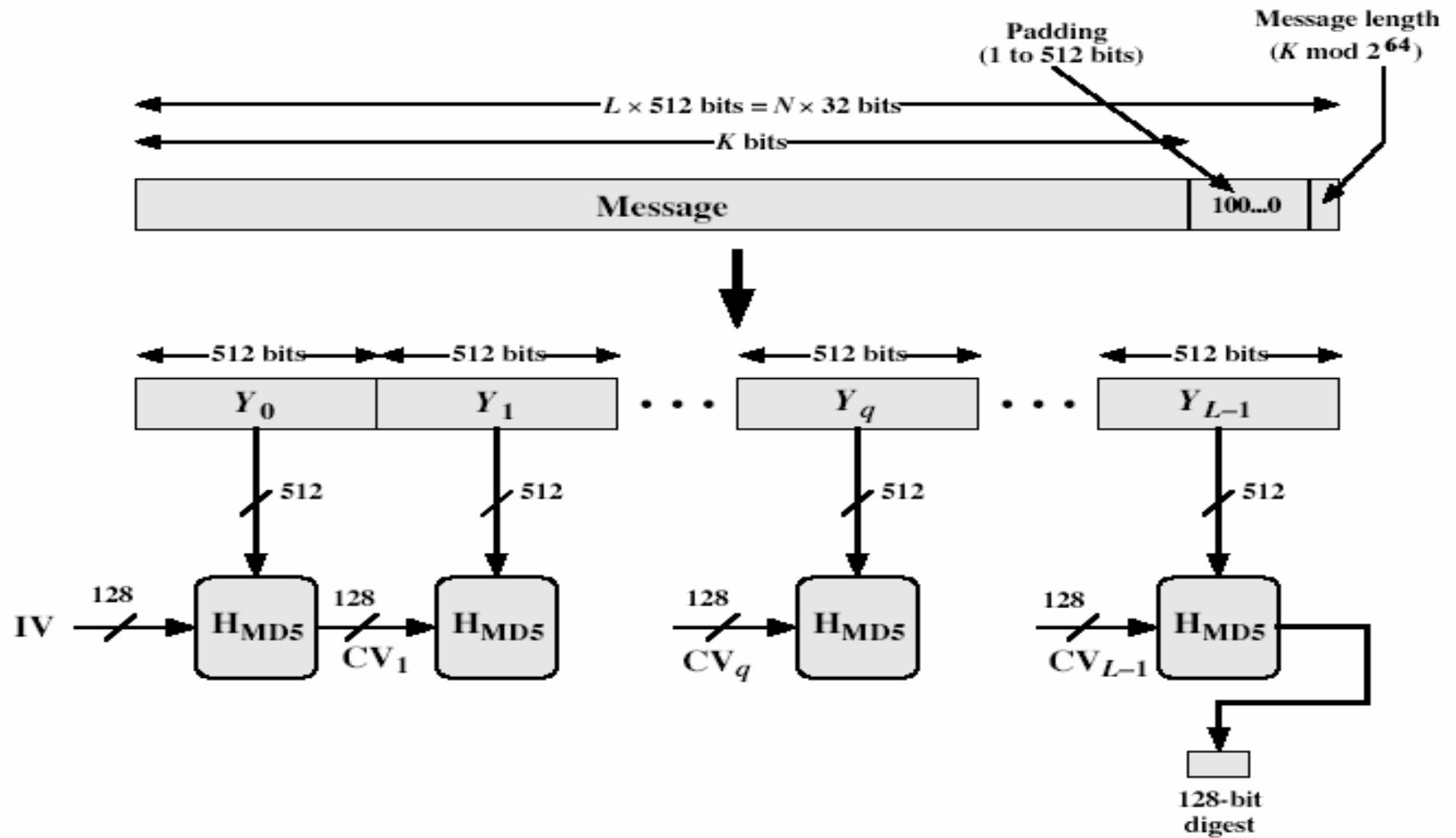
MD5 Overview

- La lunghezza del messaggio è resa pari a 448 mod 512
- Viene poi aggiunta una stringa di 64 bit, contenente la lunghezza mod 2^{64} del messaggio originale (si parte dal LSB; si ottiene così una stringa multipla di 512)
- Vengono inizializzati i 4 buffer da 32 bit l'uno che conterranno poi l'uscita. I valori di inizializzazione sono
 - A=67452301, B=EFCDAB89, C=98BADCFE
D=10325476

MD5 Overview

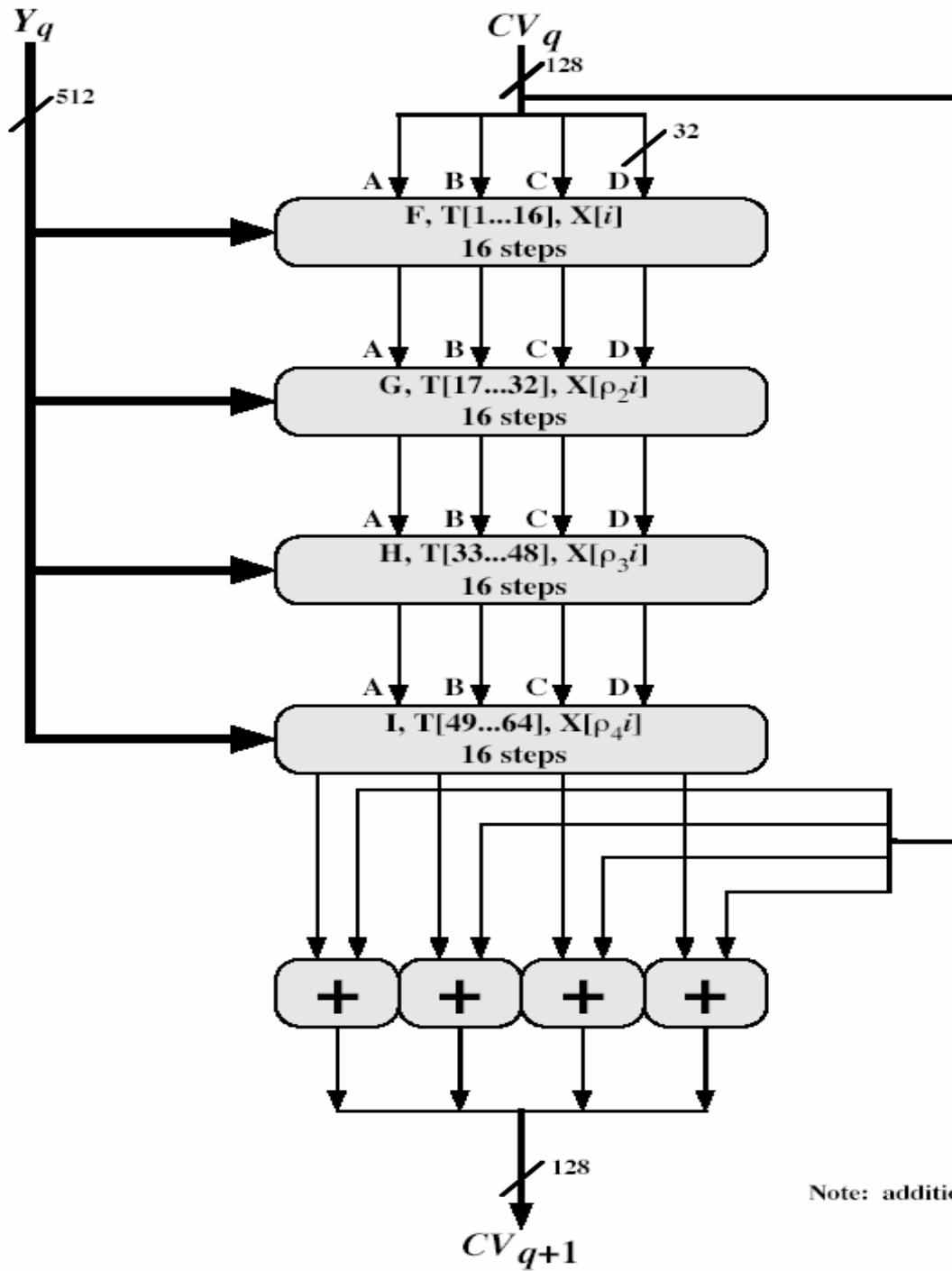
- Elabora il messaggio in blocchi da 512 bit
 - utilizza 4 fasi, in cui sono svolte operazioni sul generico blocco di messaggio e sul buffer
- Il codice hash è il contenuto dei buffer alla fine dell'algoritmo

MD5 Overview



La funzione H_{MD5}

- Ciascuna fase ha 16 passi del tipo:
$$a = b + ((a+g(b,c,d)+X[k]+T[i]) \ll s)$$
$$k=1, \dots, 16$$
- a,b,c,d sono i 4 buffer, ma utilizzati in varie combinazioni
 - In ogni passo, solo un buffer è aggiornato
 - Dopo 16 passi ogni buffer è aggiornato 4 volte
- g(b,c,d) è una funzione non lineare diversa in ogni fase (F,G,H,I)
- T[i] è un valore costante derivato dalla funzione seno

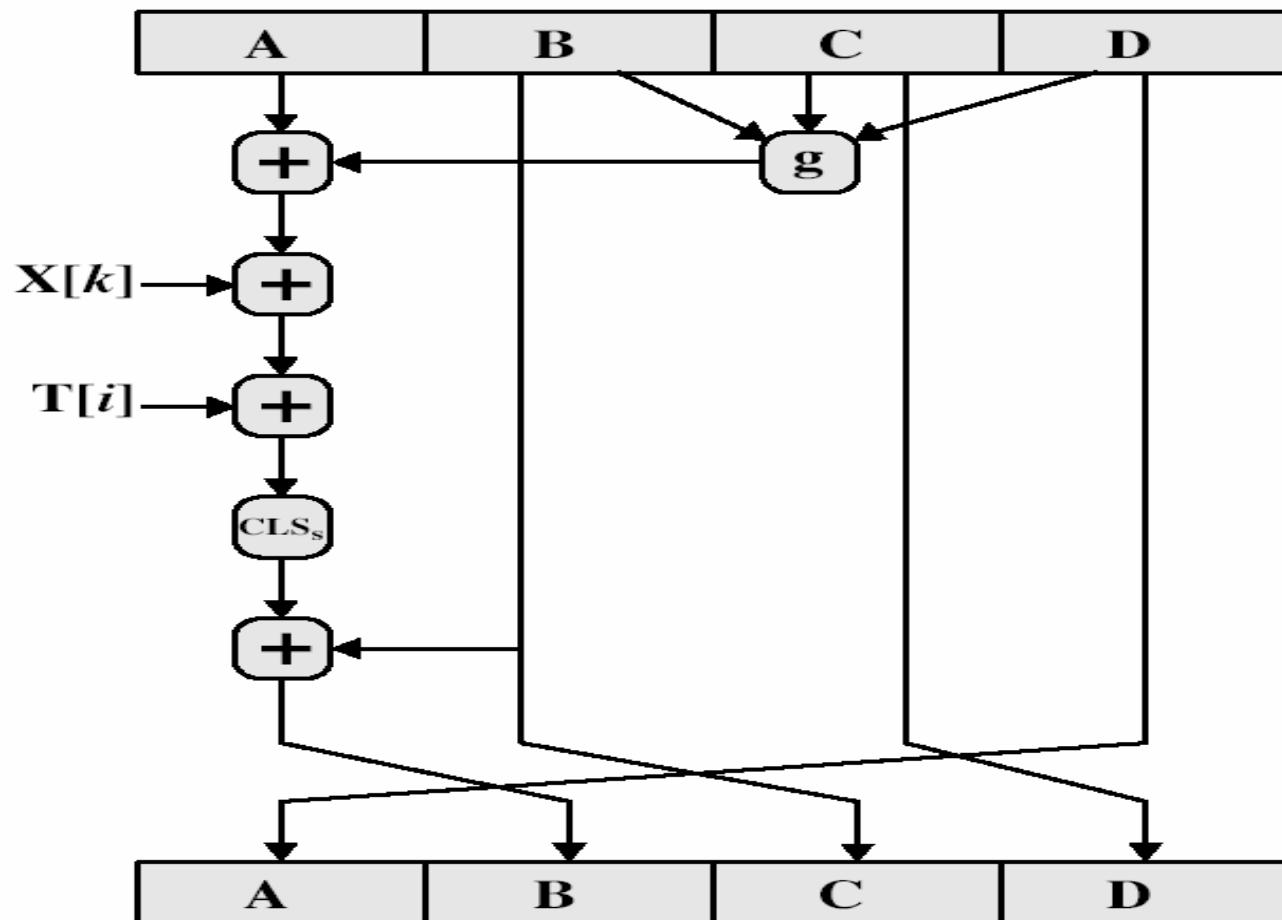


La Funzione H_{MD5}

$$\begin{aligned} p_2(i) &= (1+5i) \bmod 16 \\ p_3(i) &= (5+3i) \bmod 16 \\ p_4(i) &= 7i \bmod 16 \end{aligned}$$

Note: addition (+) is mod 2^{32}

Il generico passo



MD5: Le funzioni F G H I

Table 12.1 Key Elements of MD5

(a) Truth table of logical functions

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

MD4

- È il precursore di MD5
- Produce anch'esso un codice hash di 128 bit
- ha 3 fasi di 16 passi (invece che 4 come in MD5)
- Obiettivi di progetto (simili a quelli di MD5):
 - Resistenza alle collisioni (ovvero difficoltà nel trovare messaggi collidenti)
 - Sicurezza implicita, non dipendente da problemi computazionalmente complicati
 - Velocità, semplicità e compattezza
 - Utilizzo dell'architettura little endian (byte meno significativo nel byte con l'indirizzo più basso, è utilizzata nei PC)

MD4 versus MD5

- MD4 usa 3 fasi invece che 4 fasi
- In MD5 viene utilizzata una costante additiva $T[i]$ diversa per ciascuno dei 64 passi; MD4 usa invece una costante fissa

La forza di MD5

- Ogni bit dell'hash MD5 dipende da tutti i bit del messaggio
- Rivest sostiene nel documento RFC che MD5 è il codice hash a 128 bit più resistente possibile
- Sono stati pubblicati tuttavia vari tipi di attacchi:
 - Berson 92 ha dimostrato che è possibile trovare con l'analisi differenziale messaggi che producono lo stesso output per ciascuna fase (non è stato tuttavia in grado di estendere il risultato all'insieme delle 4 fasi)

La forza di MD5

- Boer & Bosselaers 93 si sono accorti che, per un messaggio lungo 512 bit, due diverse inizializzazioni dei buffer ABCD davano luogo allo stesso codice hash (pseudocollisione)
- Dobbertin 96 ha creato una collisione su messaggi lunghi 512 bit nel caso in cui i buffer ABCD sono azzerati
- In conclusione, MD5 è apparso in fin dei conti vulnerabile!

Secure Hash Algorithm (SHA-1)

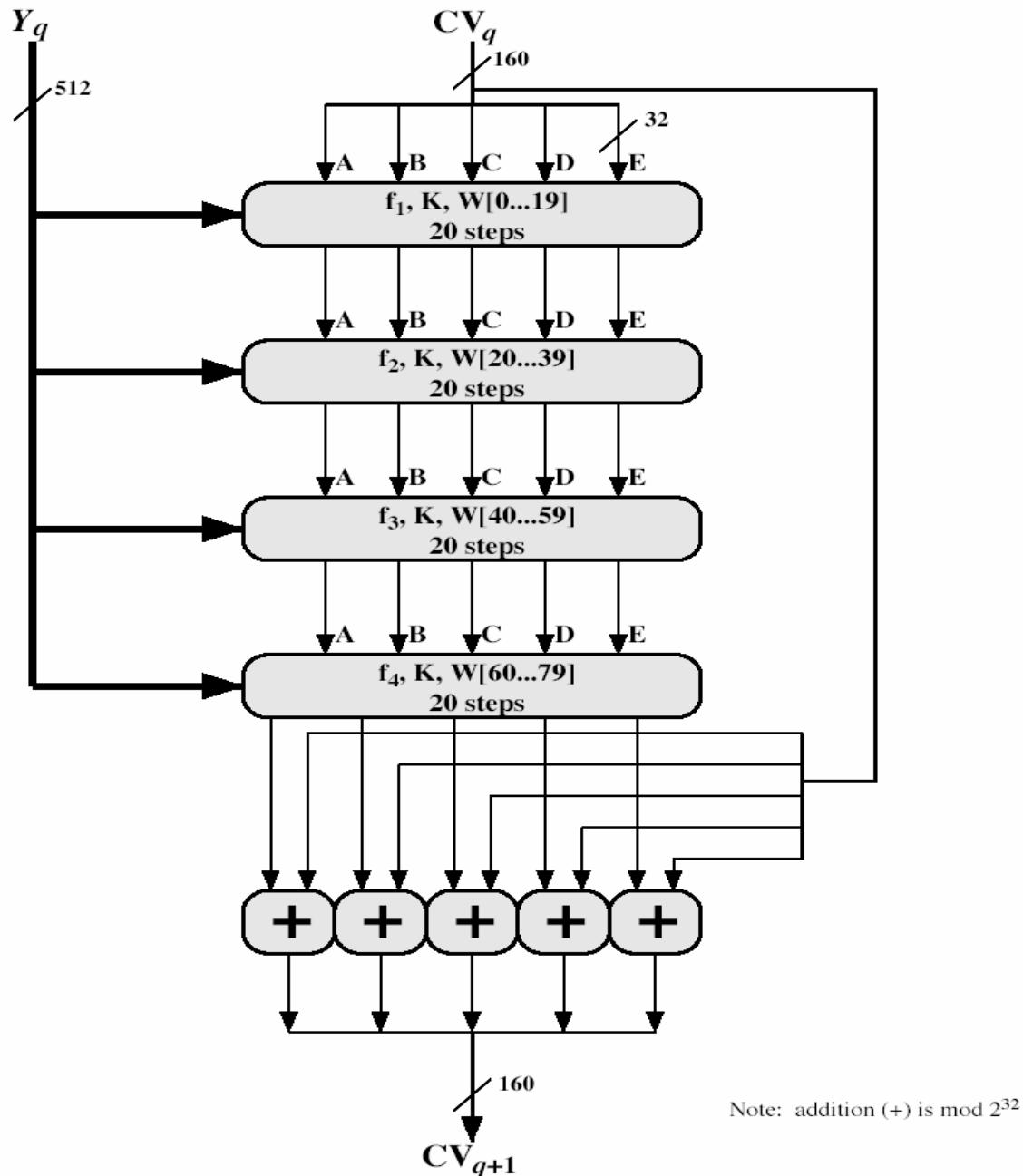
- SHA fu sviluppato dal NIST e dall' NSA nel 1993; fu poi rivisto nel 1995 e chiamato SHA-1
- US standard, usato con la tecnica di firma digitale DSA
 - Lo standard è FIPS 180-1 1995; compare anche come RFC3174
 - N.B. l'algoritmo è detto SHA, lo standard è detto SHS (Secure hash standard)
- Produce codici hash di 160 bit
- È attualmente l'algoritmo hash maggiormente preferito
- È basato sulla struttura di MD5

SHA-1 Overview

- Il messaggio viene allungato in modo tale che la sua lunghezza sia pari a 448 mod 512
- Sono poi aggiunti 64 bit che rappresentano la lunghezza, modulo 2^{64} , del messaggio originale
- I 5 buffer (A,B,C,D,E) – totale 160 bit – sono inizializzati a
(67452301,efcdab89,98badcfe,10325476,c3d2e1f0)

SHA-1 Overview

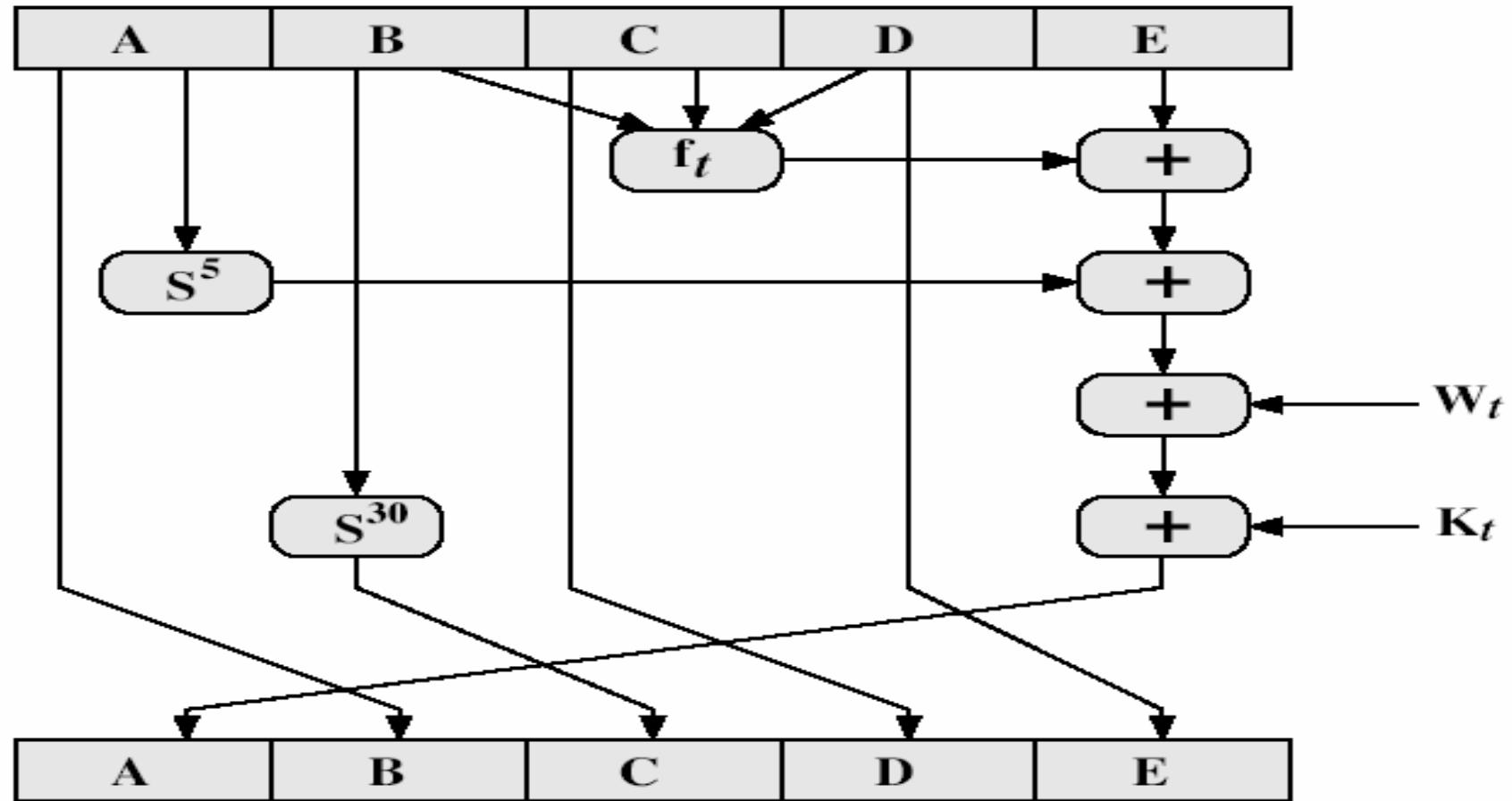
- Il messaggio è elaborato in blocchi di 512 bit (16 word da 32 bit)
- Vi sono 4 fasi da 20 passi
 - le 16 word sono espanso a 80 word tramite operazioni di miscelazione e duplicazione
 - L'output è sommato all'input per ottenere il nuovo valore dei buffer
- Il codice hash è il valore finale del buffer



Elaborazione
SHA-1 di un
singolo blocco
da 512 bit

**Figure 12.5 SHA-1 Processing of a Single 512-bit Block
(SHA-1 Compression Function)**

SHA-1



Funzionamento di SHA-1

- Nel grafico precedente....
 - a,b,c,d,e sono le 5 word del buffer
 - t è il numero di passo
 - $f(t, B, C, D)$ è la funzione non lineare della fase in esame
 - w_t è una parola di 32 bit derivata dal messaggio
 - K_t è un valore costante ottenuto prendendo alcune cifre decimali di opportuni numeri irrazionali

Le funzioni logiche di SHA-1

Table 12.2 Truth Table of Logical Functions for SHA-1

B	C	D	f_{0..19}	f_{20..39}	f_{40..59}	f_{60..79}
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

MD5 – SHA-1: Confronto

- L'attacco a forza bruta è più complicato (160 bit invece che 128)
- Al momento, non sono noti attacchi criptoanalitici verso SHA-1
- SHA-1 è leggermente più lento (80 passi invece che 64)
- Entrambi gli algoritmi sono semplici da descrivere e da implementare
- SHA-1 è ottimizzato per l'architettura big endian (al contrario di MD5)

Revisione di SHS

- Nel 2002, il NIST ha emesso una revisione di SHS tramite il documento FIPS 180-2
- Sono aggiunti 3 nuovi algoritmi hash:
 - SHA-256, SHA-384, SHA-512
- La struttura di tali algoritmi è simile a quella di SHA-1
- Di conseguenza, anche l'analisi di tali algoritmi si basa sulle stesse tecniche utilizzate per SHA-1

Table 12.3 Comparison of SHA Properties

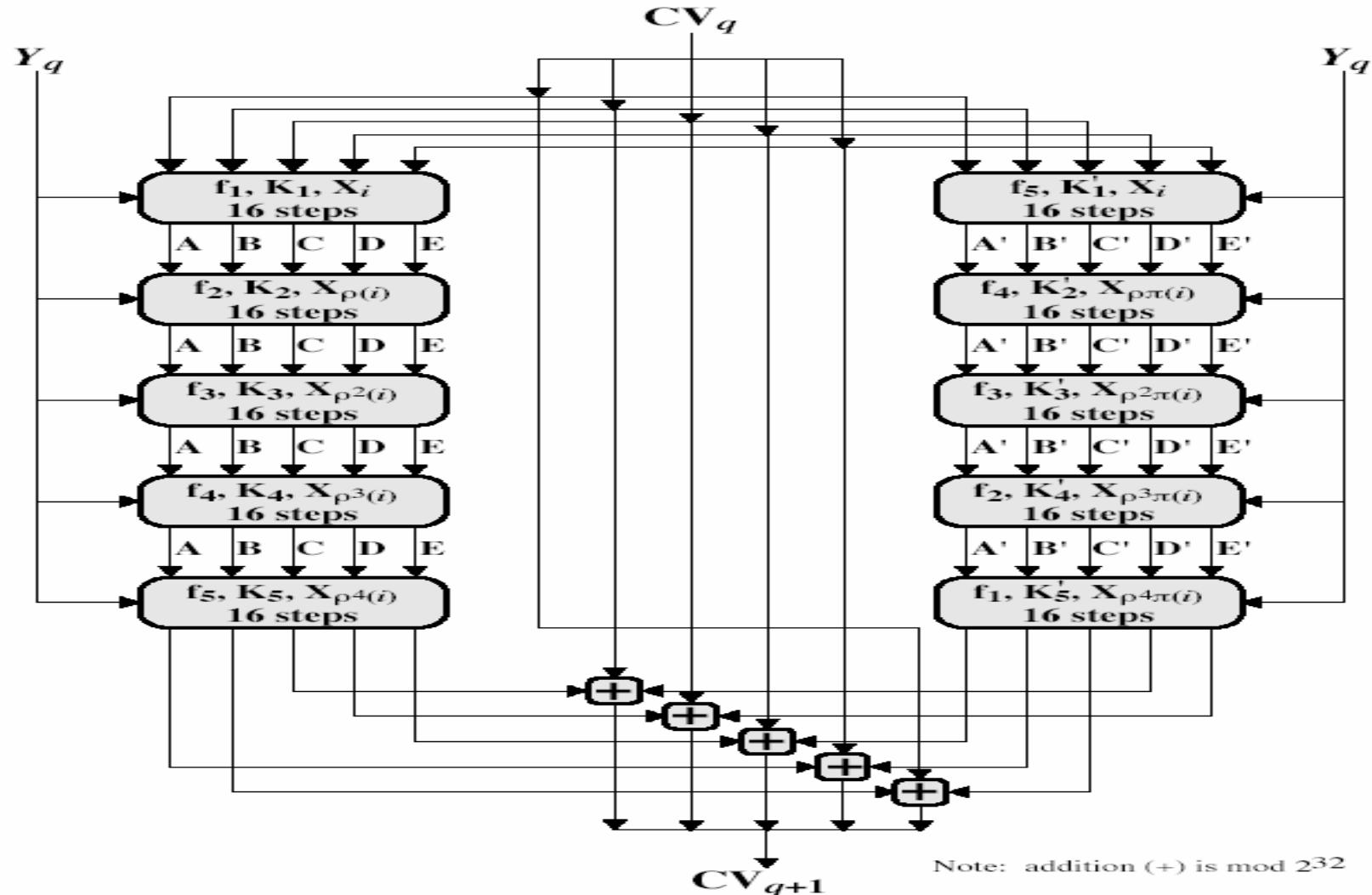
	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	80	80	80
Security	80	128	192	256

- Notes:
1. All sizes are measured in bits.
 2. Security refers to the fact that a birthday attack on a message digest of size n produces a collision with a workfactor of approximately $2^{n/2}$.

RIPEMD-160

- RIPEMD-160 fu sviluppato in Europa nell'ambito di un progetto di ricerca europeo
- Gli ideatori sono i ricercatori che concepirono gli attacchi a MD4/MD5
- È l'evoluzione di un precedente algoritmo a 128 bit, rivelatosi vulnerabile
- È simile a MD5/SHA
- usa 2 cascate parallele di 5 fasi da 16 passi
- Il codice hash prodotto è di 160 bit
- È più lento, ma forse più sicuro, di SHA-1
- Architettura little-endian

Struttura di una fase di RIPEMD-160



Codici MAC derivati da algoritmi hash

- Abbiamo visto come DES possa essere utilizzato per creare un codice MAC
- Recentemente, ci si è interessati alla creazione di codici MAC a partire da algoritmi hash
 - Gli algoritmi hash sono infatti generalmente più veloci
 - Non limitati da divieti di esportazione come gli algoritmi crittografici

Codici MAC derivati da algoritmi hash

- Una possibilità è di includere la chiave nel messaggio e poi calcolare il codice hash
- Ovvero:
$$\text{KeyedHash} = \text{Hash}(\text{Key} \mid \text{Message})$$
 - Tale schema presenta delle vulnerabilità
- Tale approccio ha poi portato allo sviluppo di HMAC

HMAC

- È un internet standard RFC2104
- È obbligatorio per la sicurezza IP e viene usato in protocolli internet come SSL
- Obiettivi progettuali di HMAC:
 - Utilizzo delle funzioni hash liberamente disponibili e presenti nelle librerie software
 - Possibilità di facile sostituzione delle funzioni hash utilizzate
 - Preservare la robustezza degli algoritmi hash utilizzati
 - Utilizzare e gestire le chiavi in modo semplice
 - Avere buona resistenza all'analisi crittografica

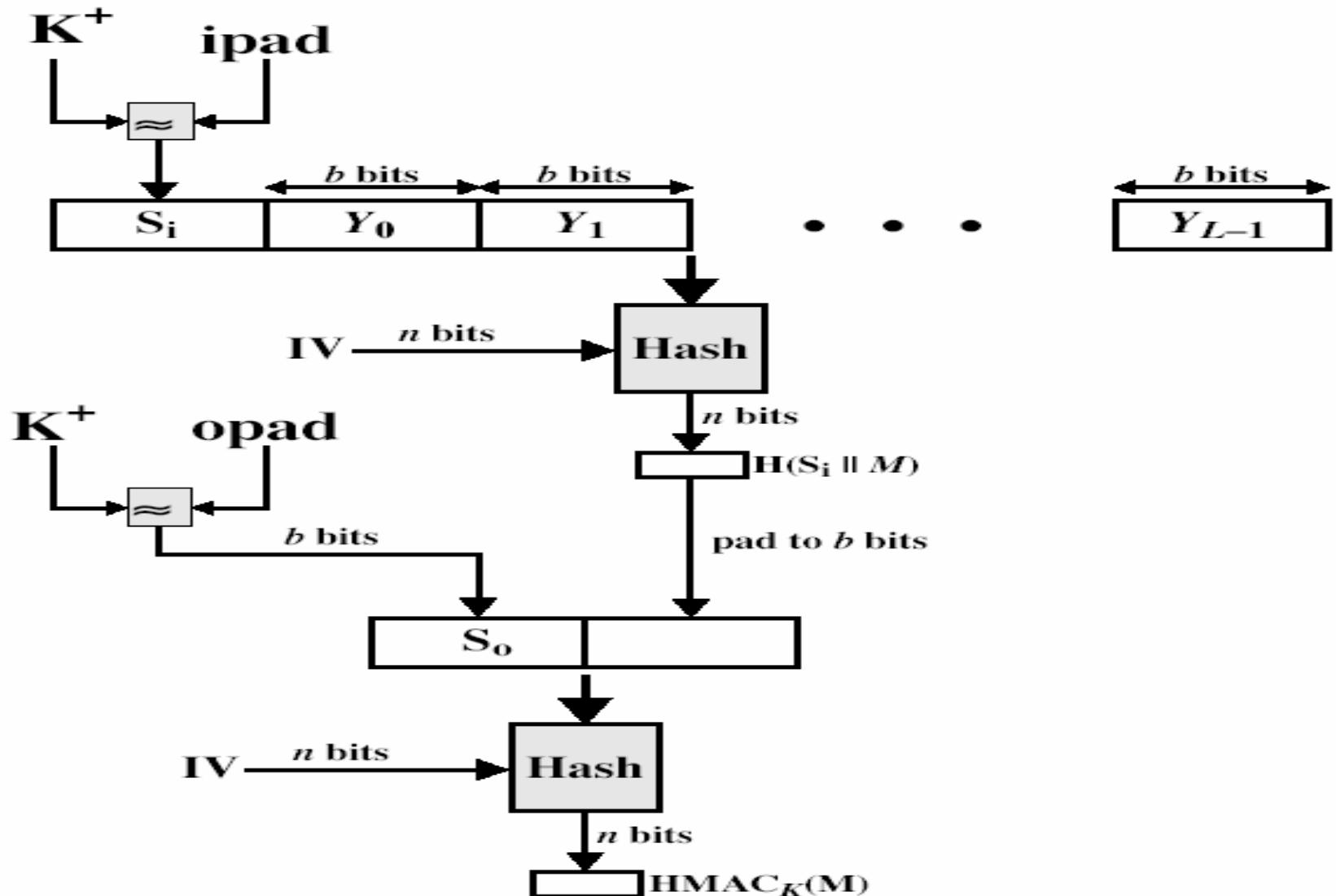
HMAC

- Utilizza un algoritmo hash:

$$\text{HMAC}_K = \text{Hash}[(K^+ \text{ XOR } \text{opad}) \parallel \\ \text{Hash}[(K^+ \text{ XOR } \text{ipad}) \parallel M]]$$

- where K^+ è la chiave zero-padded a sinistra in modo che sia lunga b bit (b è la lunghezza del blocco elaborato dalla funzione hash)
- opad=5C (ripetuto $b/8$ volte), ipad=36 (ripetuto $b/8$ volte)
- MD5, SHA-1, RIPEMD-160 possono essere utilizzati

Funzionamento di HMAC



Firme digitali e protocolli di autenticazione

Firme Digitali

- Come detto più volte, la firma digitale ha la capacità di
 - Verificare l'autenticità del mittente e individuare la data e l'ora della firma
 - Verificare l'integrità del messaggio
 - Permettere l'esibizione della firma ad un'autorità estranea per risolvere dispute e contenziosi

Requisiti di una firma digitale

- Deve dipendere dal messaggio che si sta firmando
- Deve utilizzare informazioni specifiche del mittente
 - Per evitare modifiche e ripudiabilità dell'origine
- Deve essere alquanto facile da produrre
- Deve essere facile da riconoscere e verificare
- Deve essere computazionalmente impossibile
 - Creare un nuovo messaggio con una firma a disposizione
 - Creare una nuova firma per un dato messaggio fraudolento
- Deve essere facile memorizzare le firme per verifiche future

Firma digitale diretta

- Coinvolge solo mittente e destinatario
- Ne abbiamo già parlato, richiede l'uso della crittografia a chiave pubblica
- Il mittente firma l'intero messaggio o un codice hash con la sua chiave privata
- La sicurezza dipende sull'inviolabilità della chiave privata del mittente

Firma digitale arbitrata



Firma digitale arbitrata

- Fa affidamento su un arbitro A che verifica i messaggi prima che giungano al destinatario
 - L'arbitro verifica la firma su ogni messaggio
 - Dopodichè lo data e lo invia al destinatario
- Al solito, richiede un grande livello di **fiducia** nell'arbitro
- Può essere implementata sia con crittografia simmetrica che asimmetrica
- L'arbitro può avere o non avere accesso ai messaggi

Firma digitale arbitrata

(a) Conventional Encryption, Arbiter Sees Message

(1) $X \rightarrow A: M \parallel E_{K_{xa}}[ID_X \parallel H(M)]$

(2) $A \rightarrow Y: E_{K_{ay}}[ID_X \parallel M \parallel E_{K_{xa}}[ID_X \parallel H(M)] \parallel T]$

(b) Conventional Encryption, Arbiter Does Not See Message

(1) $X \rightarrow A: ID_X \parallel E_{K_{xy}}[M] \parallel E_{K_{xa}}[ID_X \parallel H(E_{K_{xy}}[M])]$

(2) $A \rightarrow Y: E_{K_{ay}}[ID_X \parallel E_{K_{xy}}[M] \parallel E_{K_{xa}}[ID_X \parallel H(E_{K_{xy}}[M])] \parallel T]$

(c) Public-Key Encryption, Arbiter Does Not See Message

(1) $X \rightarrow A: ID_X \parallel E_{KR_x}[ID_X \parallel E_{KU_y}(E_{KR_x}[M])]$

(2) $A \rightarrow Y: E_{KR_a}[ID_X \parallel E_{KU_y}[E_{KR_x}[M]] \parallel T]$

Protocolli di Autenticazione

- Utilizzati per convincere i soggetti della comunicazione dell'identità reciproca e per scambiarsi le chiavi di sessione
- Possono essere monodirezionali o reciproci
- Questioni cruciali sono
 - confidenzialità – per proteggere le chiavi di sessione
 - puntualità – per prevenire gli attacchi a replay

Uso della crittografia simmetrica: Needham-Schroeder Protocol

- Si fa affidamento su un KDC
- La sessione tra A e B è mediata dal KDC
- Il protocollo è il seguente:
 1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks] \parallel ID_A]$
 3. A→B: $E_{Kb}[Ks] \parallel ID_A$
 4. B→A: $E_{Ks}[N_2]$
 5. A→B: $E_{Ks}[f(N_2)]$

Protocollo Needham-Schroeder

- Utilizzato per distribuire in modo sicuro una nuova chiave di sessione tra A e B
- È vulnerabile a un attacco a replay se una vecchia chiave di sessione è stata violata
 - Il messaggio 3 può essere reinviato convincendo B che sta comunicando con A
- Per porre rimedio c'è bisogno di
 - timestamps
 - Utilizzo di un nonce extra

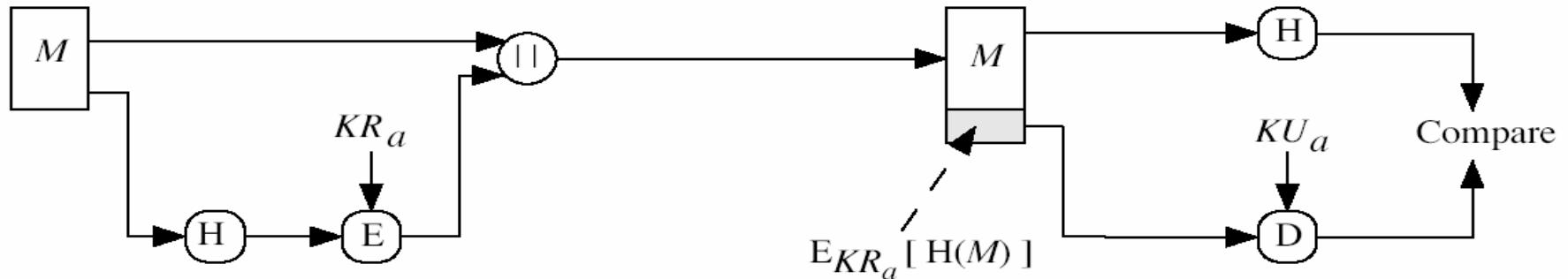
Uso della crittografia asimmetrica con Authentication Server

- Denning 81 ha proposto il seguente schema:
 1. A→AS: $ID_A \parallel ID_B$
 2. AS→A: $E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T]$
 3. A→B: $E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T] \parallel E_{KUb}[E_{KRas}[K_s \parallel T]]$
- Si noti che la chiave di sessione è scelta da A, quindi il livello di fiducia da riporre sull'AS è minore
- I timestamps prevengono gli attacchi a replay ma richiedono una sincronizzazione dei clock

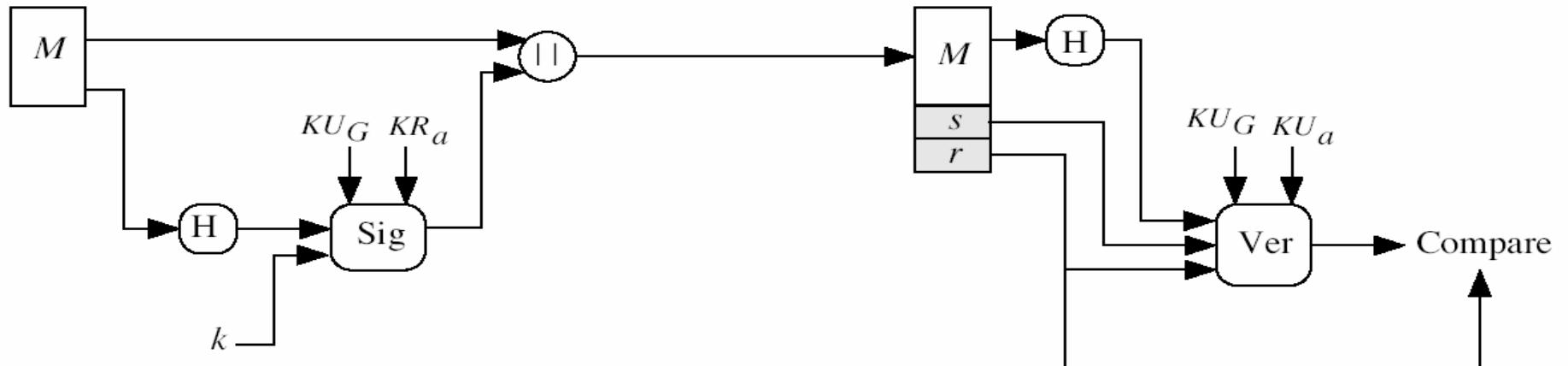
Digital Signature Standard (DSS)

- E' lo schema ufficiale di firma digitale approvato dagli USA (FIPS 186)
- Utilizza l'algoritmo SHA
- Progettato da NIST & NSA nei primi anni '90
- DSS è lo standard, DSA è l'algoritmo
- La firma è lunga 320 bit, ma con una sicurezza dell'ordine di 512-1024
- La forza dell'algoritmo si basa sulla difficoltà di calcolo del logaritmo discreto

RSA vs. DSS



(a) RSA Approach



(b) DSS Approach

Generazione della chiave in DSA

- Vi sono anzitutto dei valori pubblicamente condivisi (p, q, g):
 - $p \approx 2^L$
 - ove L è multiplo di 64 e varia tra 512 e 1024
 - q , è un numero primo di 160 bit e fattore di $p-1$
 - $g = h^{(p-1)/q}$
 - ove $h < p-1$, $h^{(p-1)/q} \pmod{p} > 1$
- Ogni utente sceglie la sua chiave privata x e calcola quella pubblica y :
 - si sceglie quindi $x < q$
 - e si calcola $y = g^x \pmod{p}$

DSA: Creazione della firma

- per **firmare** un messaggio M , il mittente:
 - Genera una chiave aleatoria k , $k < q$
(N.B. k deve essere aleatoria, usata una sola volta e poi distrutta)
 - Calcola poi i parametri di firma:
$$r = (g^k \pmod p) \pmod q$$
$$s = k^{-1} (\text{SHA}(M) + x \cdot r) \pmod q$$
 - Invia la firma (r, s) insieme al messaggio M

DSA: Verifica della Firma

- ... avendo ricevuto M & la firma (r, s)
- per **verificare** una firma, il destinatario calcola

$$w = s^{-1} \pmod{q}$$

$$u_1 = (\text{SHA}(M) \cdot w) \pmod{q}$$

$$u_2 = (r \cdot w) \pmod{q}$$

$$v = (g^{u_1} \cdot y^{u_2} \pmod{p}) \pmod{q}$$

- se $v=r$ allora la firma è verificata
- Vediamo perchè funziona...

DSA: Proof

- Se la firma è autentica, deve essere

$$\text{SHA}(M) = -xr + ks \pmod{q}$$

- Premoltiplicando per w . . .

$$w\text{SHA}(M) + wxr = wks \pmod{q}$$

$$w\text{SHA}(M) + wxr = k \pmod{q}$$

$$u_1 + xu_2 \pmod{q} = k \pmod{q} = k$$

(nell'ultimo passaggio si è tenuto conto del fatto che $k < q$)

DSA: Proof

- Ora, da tale relazione si ha

$$g^{(u_1 + xu_2) \pmod{q}} = g^k$$

- D'altra parte, $g^q = h^{p-1} \pmod{p-1}$ per il th. di Fermat, quindi

$$g^{(u_1 + xu_2)} = g^k$$

$$g^{u_1} (g^x)^{u_2} = g^k$$

- E, prendendo tale relazione mod p mod q si ha:

$$g^{u_1} (g^x)^{u_2} \pmod{p} \pmod{q} = g^k \pmod{p} \pmod{q}$$

$$g^{u_1} \cdot y^{u_2} \pmod{p} \pmod{q} = g^k \pmod{p} \pmod{q}$$

DSA: Proof

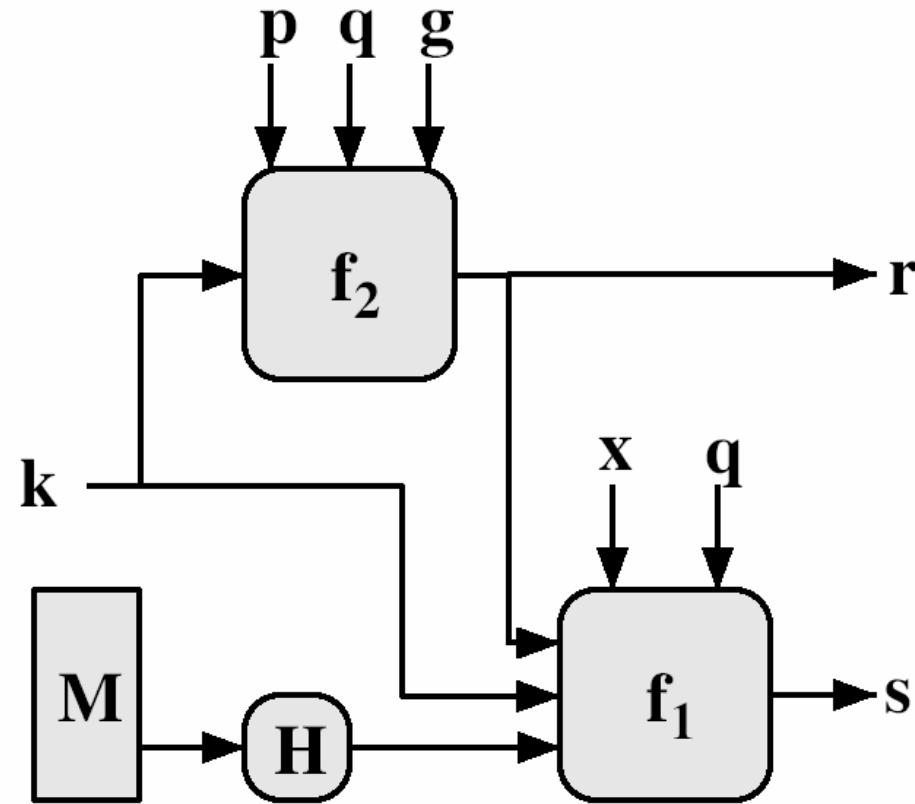
- In definitiva, abbiamo trovato che

$$g^{u_1}y^{u_2} \bmod p \bmod q = g^k \bmod p \bmod q$$

$$\begin{matrix} & \\ v & = & r \end{matrix}$$


- Altrimenti detto, lo schema funziona!

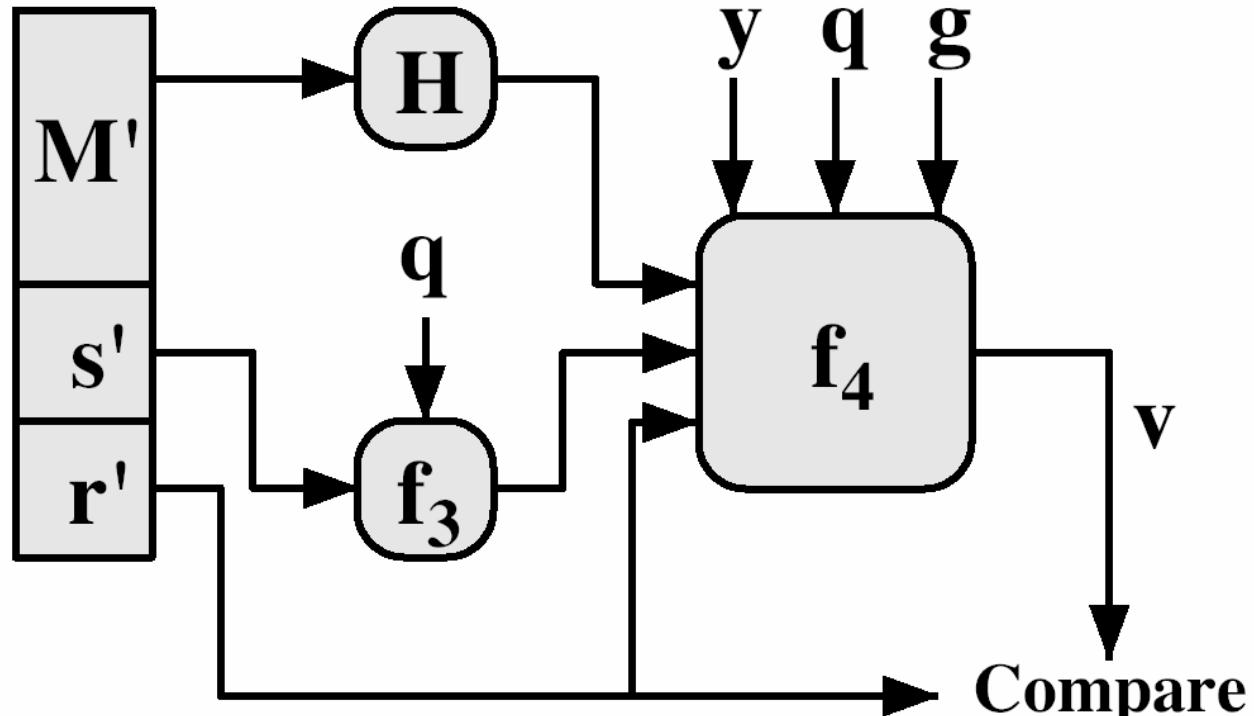
DSA: La firma



$$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$$

$$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$$

DSA: La verifica



$$w = f_3(s', q) = (s')^{-1} \bmod q$$

$$v = f_4(y, q, g, H(M'), w, r')$$

$$= ((g(H(M')w) \bmod q \cdot y^{r'} w \bmod q) \bmod p) \bmod q$$

La sicurezza della posta elettronica

PGP (Pretty Good Privacy),
S/MIME

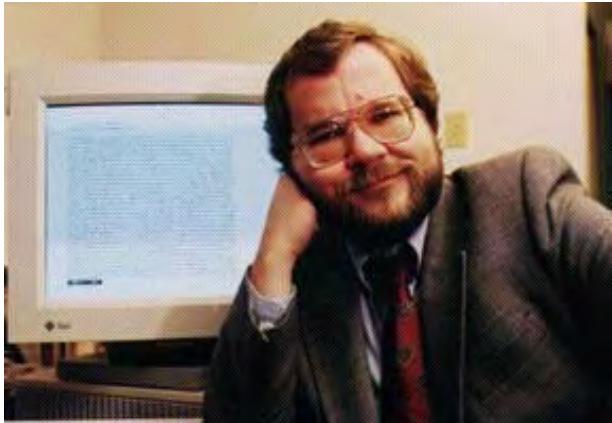
L'e-mail...

- Il servizio di e-mail è una delle applicazioni più largamente usata dagli utenti di internet
- Al momento, le e-mail comunemente utilizzate non sono crittografate
 - Possono quindi essere captate durante la loro trasmissione sulla rete
 - O, anche, da utenti privilegiati del mailserver di destinazione

Requisiti di sicurezza desiderati

- riservatezza
 - Protezione della privacy
- autenticazione
 - Del mittente del messaggio
- integrità
 - Protezioni da modifiche
- Non-ripudiabilità dell'origine
 - Il mittente non deve poter negare l'invio del messaggio

Pretty Good Privacy (PGP)



Philip R. Zimmermann

(ideatore e creatore di PGP)

www.philzimmermann.com

Pretty Good Privacy (PGP)

- Rappresenta uno standard “de facto” per l’e-mail sicura
- Sviluppato da Phil Zimmermann
- Egli impiegò i migliori algoritmi crittografici
- ... e li integrò in un unico programma semplice e disponibile per una varietà di sistemi operativi: Unix, PC, Macintosh e Amiga
- Originariamente era gratuito, oggi lo è ancora, ma sono disponibili anche delle versioni commerciali

PGP

- Phil Zimmermann ha reso la documentazione ed il codice sorgente dell'applicazione liberamente disponibile in rete
- ... ha stretto accordi con la Viacrypt (ora Network Associates) per fornire una versione commerciale pienamente compatibile e a basso costo di PGP
- PGP è ora uno standard per internet (RFC 3156)

I servizi di PGP

- PGP comprende 5 servizi
 - Autenticazione
 - Segretezza
 - Compressione
 - Compatibilità con la posta elettronica
 - Segmentazione

PGP: Autenticazione

1. Il mittente crea il messaggio
2. L'algoritmo SHA-1 viene utilizzato per generare un hash di 160 bit
3. Viene poi usato DSS o RSA per criptare il codice hash
4. Al solito, il messaggio è accettato come autentico solo se l'hash generato in ricezione coincide con quello inviato

PGP: Autenticazione

- Le firme non sono sempre allegate al messaggio. A volte vengono inviate separatamente.
- Ciò è utile
 - Per rivelare eventuali virus che si sono auto-allegati all'e-mail
 - Quando un messaggio deve essere firmato da più parti
 - Per conservare una registrazione separata delle firme dei messaggi ricevuti

PGP: Confidenzialità

1. Il mittente genera un messaggio e una chiave random di 128 bit utilizzata solo per questo messaggio
2. Si utilizza l'algoritmo CAST-128, IDEA o 3DES
3. La chiave di sessione è criptata con RSA (con chiave pubblica del destinatario e posta in attachment)
(in alternativa a RSA, PGP prevede anche l'utilizzo di una variante di Diffie-Hallmann – crittografia di ElGamal – che permette di realizzare la crittografia del messaggio)

PGP: Confidenzialità e Autenticazione

- Vengono implementati entrambi i servizi precedenti su un unico messaggio
 - Si crea la firma e la si allega al messaggio
 - Il messaggio e la firma sono entrambi criptati
 - Viene poi allegata la chiave di sessione, criptata con RSA
(questa sequenza è preferibile a quella che prevede prima crittografia e poi firma)

PGP: Compressione

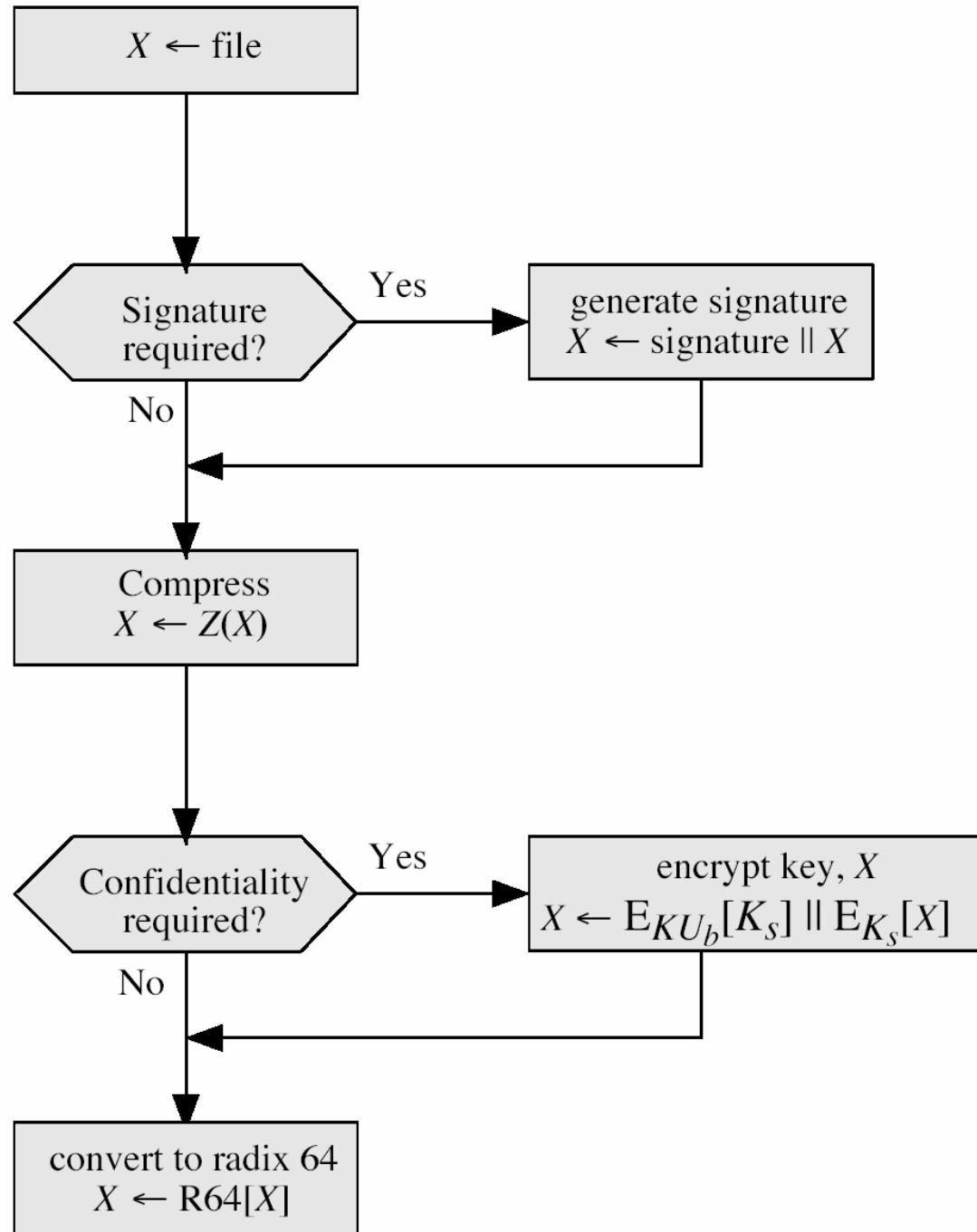
- PGP comprime in maniera automatica i messaggi dopo che sono stati firmati ma prima della criptazione
 - Si può quindi conservare il messaggio non compresso e la sua firma per eventuali verifiche
 - Inoltre l'algoritmo di compressione non è deterministico, quindi è meglio prima firmare e poi comprimere
- Si utilizza l'algoritmo ZIP (Lempel-Ziv)
- L'applicazione della crittografia dopo la compressione migliora la sicurezza del sistema

PGP: Compatibilità con l'e-mail

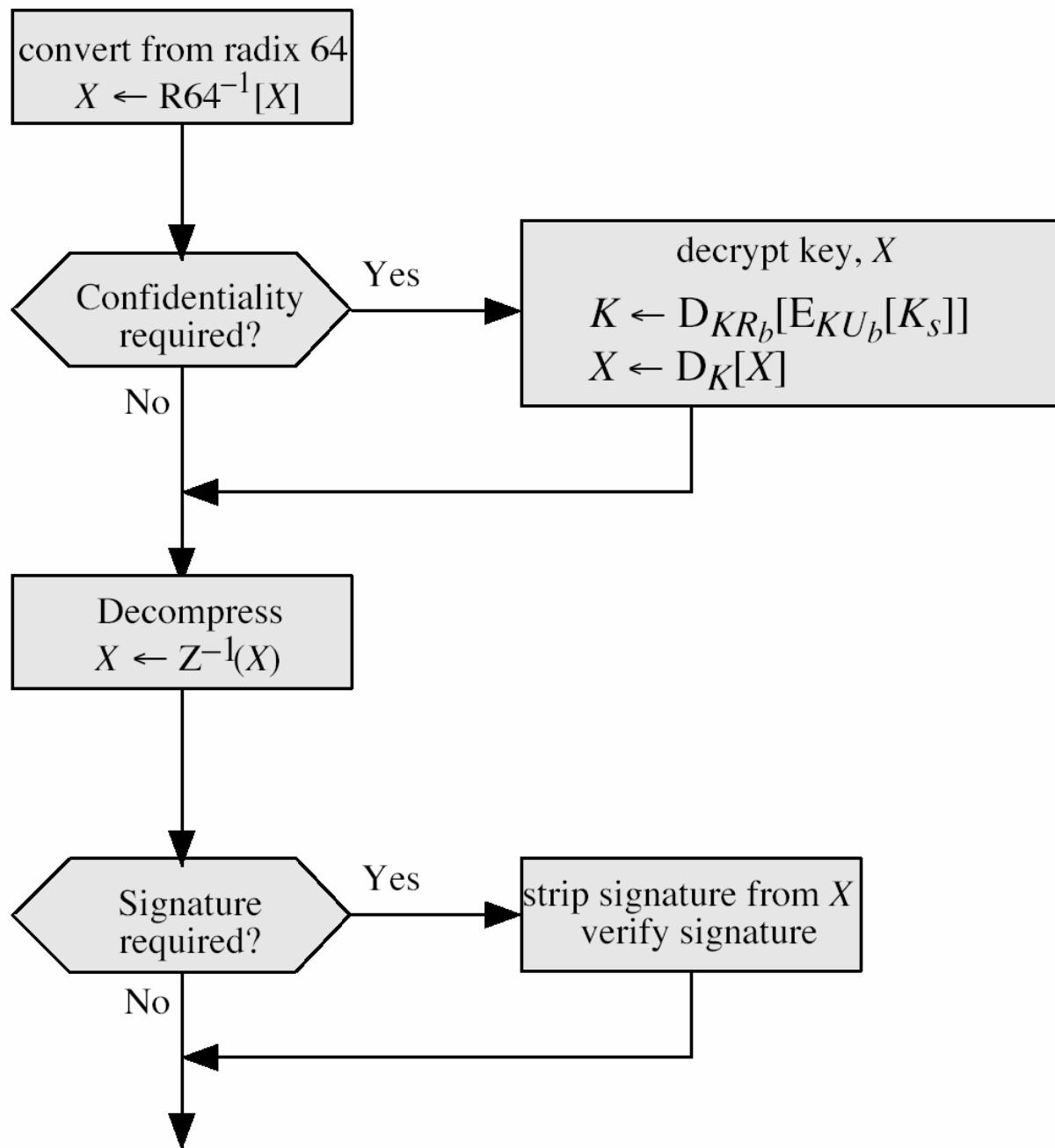
- L'output delle operazioni di firma e criptazione è un file binario
- Tuttavia l'e-mail è stata originariamente concepita per l'invio di testi
- Quindi PGP deve convertire i file binari in caratteri di testo ASCII stampabili
- Si utilizza l'algoritmo radix-64
 - Associa a 3 bytes 4 caratteri stampabili
 - Aggiunge poi un bit di parità
 - La dimensione del messaggio aumenta del 33%

PGP: Compatibilità con l'e-mail

- Tale espansione viene poi compensata tramite la funzione zip
- PGP fornisce anche un servizio di segmentazione e riassemblaggio
 - La segmentazione avviene dopo la compressione e la criptazione
 - La chiave di sessione e la firma compariranno quindi solo all'inizio del primo sottomessaggio



PGP
lato
mittente



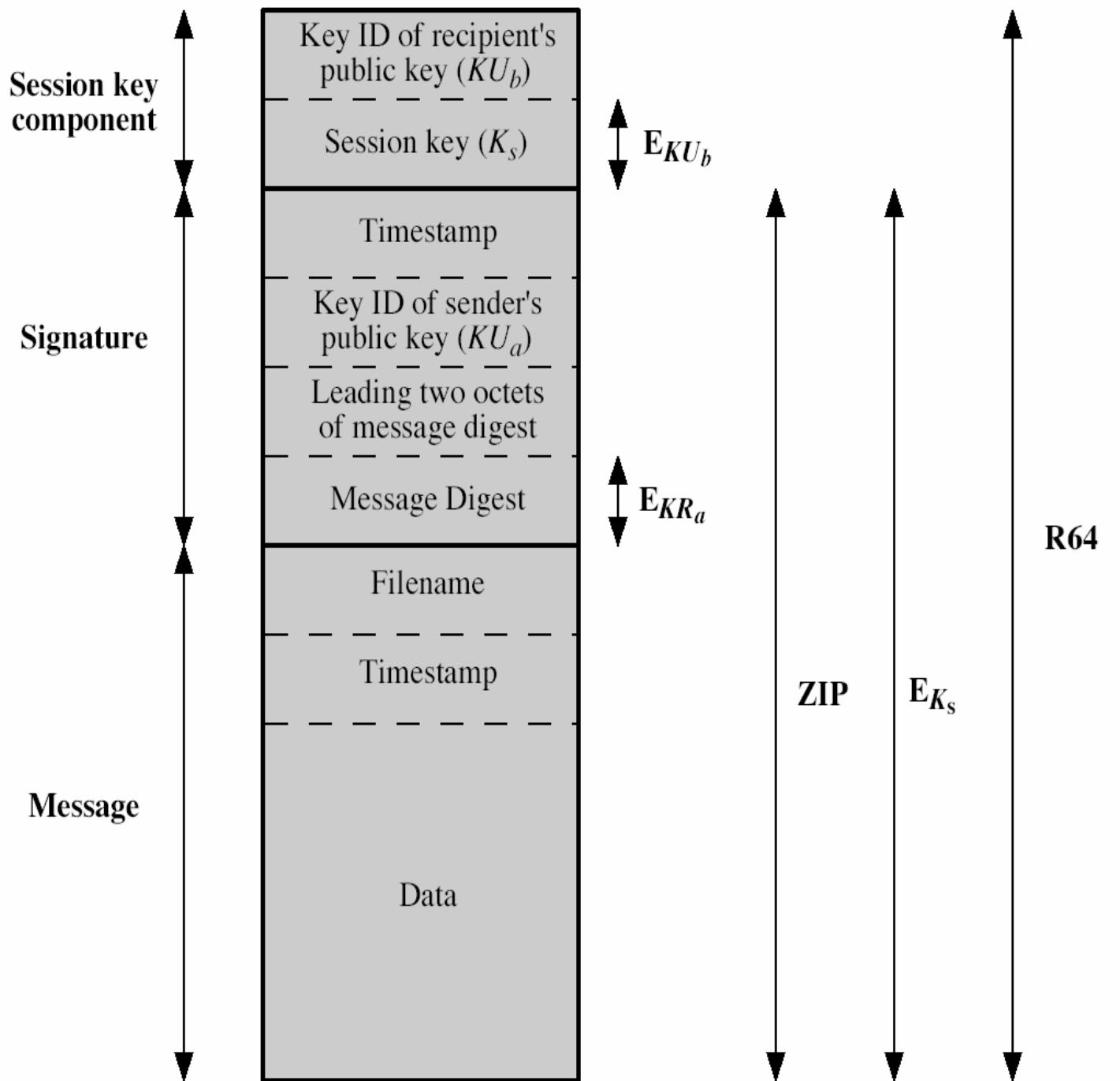
PGP
lato
ricevente

PGP: Generazione delle chiavi monouso

- Ogni messaggio è criptato con una chiave di sessione
 - Di dimensione variabile: 56-bit DES, 128-bit CAST o IDEA, 168-bit Triple-DES
- Si utilizza un algoritmo di generazione pseudocasuale ANSI X12.17
- Utilizza input aleatori quali i tasti premuti sulla tastiera e le pause tra una pressione e l'altra

PGP: Le chiavi pubbliche

- Ciascun utente può utilizzare simultaneamente più coppie (chiave pubblica, chiave privata)
- Bisogna poter identificare quale chiave viene utilizzata per criptare una data chiave di sessione
- Si utilizza quindi un identificatore di chiavi
 - Sono gli ultimi 64 bit della chiave
 - La probabilità di una collisione è molto bassa
- Tale ID viene allegato anche alla firma digitale, ovvero al codice hash criptato



PGP:
Formato
generale di
un
messaggio

PGP: I portachiavi (Key Ring)

- Ciascun utente PGP ha due portachiavi:
 - Il portachiavi delle chiavi pubbliche contiene tutte le chiavi pubbliche di utenti PGP conosciuti, ordinati a seconda del Key ID
 - Il portachiavi delle chiavi private contiene le coppie (chiave pubblica, chiave privata) di chiavi dell'utente, ordinate a seconda del Key ID e conservate in forma criptata con chiave generata, mediante un hash, da una frase segreta (passphrase)

Il portachiavi privato

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T _i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(Pi)}[KR_i]$	User <i>i</i>
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Il portachiavi delle chiavi pubbliche

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$KU_i \bmod 2^{64}$	Ku_i	trust_flag_i	User i	trust_flag_i		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = field used to index table

Spiegheremo dopo i campi “Owner Trust,” “Key legitimacy” e “Signature Trust”

PGP: Gestione delle chiavi

- Siamo arrivati al punto nodale...
.... come si gestiscono le chiavi pubbliche??
- Non si ricorre ad autorità di certificazione
- in PGP ogni utente è una sorta di “autorità di certificazione”...
 - ...può quindi “firmare” chiavi pubbliche di altri utenti a lui noti
- Altrimenti detto, i vari utenti si legittimano reciprocamente; ogni utente ha nei confronti degli altri utenti un dato livello di fiducia (*trust*)

PGP: Gestione delle chiavi

- Ogni voce del portachiavi pubblico è un certificato
- A ogni chiave è associato un campo di legittimità della chiave
- A ogni chiave sono associate anche zero o più firme
- A ciascuna firma è associato un campo di fiducia della firma
- Il campo di legittimità della chiave deriva dal campo di fiducia della firma

PGP: Gestione delle chiavi

- Vi è poi il **campo di fiducia del proprietario**, che indica l'affidabilità del proprietario nel firmare certificati
- Quindi I campi di fiducia della firma sono copie dei campi di fiducia del proprietario
- Quando un utente inserisce una nuova voce nel portachiavi pubblico, deve anche indicare la fiducia nel proprietario della chiave
- È possibile poi allegare una o più firme, con i relativi campi di fiducia
- Il valore del campo di legittimità della chiave viene calcolato sulla base dei valori presenti tra I campi di fiducia delle firme

PGP: Gestione delle chiavi

- Si forma quindi un “web of trust”
 - Si ripone fiducia nelle chiavi che sono formate da persone di cui ci fidiamo
- Nota che gli utenti possono anche revocare le loro chiavi pubbliche: Viene emesso un certificato di revoca, firmato con la chiave privata corrispondente alla chiave pubblica che si vuole revocare
- Periodicamente, PGP elabora il portachiavi pubblico per verificarne la coerenza

PGP: Gestione delle chiavi

(a) Trust Assigned to Public-Key Owner (appears after key packet; user defined)	(b) Trust Assigned to Public Key/User ID Pair (appears after User ID packet; computed by PGP)	(c) Trust Assigned to Signature (appears after signature packet; cached copy of OWNERTRUST for this signator)
<p>OWNERTRUST Field</p> <ul style="list-style-type: none"> – undefined trust – unknown user – usually not trusted to sign other keys – usually trusted to sign other keys – always trusted to sign other keys – this key is present in secret key ring (ultimate trust) <p>BUCKSTOP bit</p> <ul style="list-style-type: none"> – set if this key appears in secret key ring 	<p>KEYLEGIT Field</p> <ul style="list-style-type: none"> – unknown or undefined trust – key ownership not trusted – marginal trust in key ownership – complete trust in key ownership <p>WARNONLY bit</p> <ul style="list-style-type: none"> – set if user wants only to be warned when key that is not fully validated is used for encryption 	<p>SIGTRUST Field</p> <ul style="list-style-type: none"> – undefined trust – unknown user – usually not trusted to sign other keys – usually trusted to sign other keys – always trusted to sign other keys – this key is present in secret key ring (ultimate trust) <p>CONTIG bit</p> <ul style="list-style-type: none"> – set if signature leads up a contiguous trusted certification path back to the ultimately trusted key ring owner

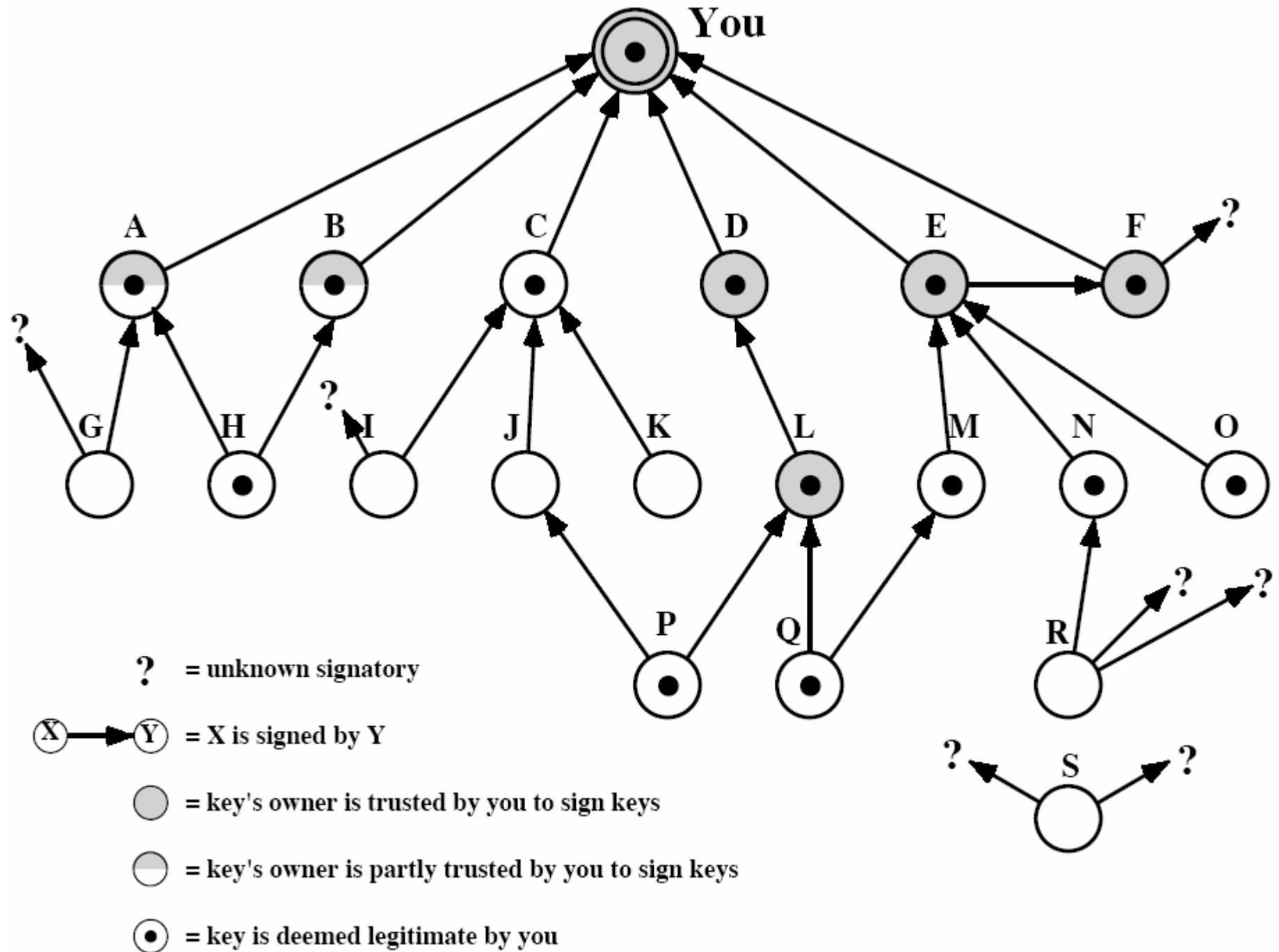


Figure 15.7 PGP Trust Model Example

S/MIME (Secure/Multipurpose Internet Mail Extensions)

- Contiene l'introduzione di principi di sicurezza allo standard di e-mail MIME
 - Il servizio di mail definito originariamente nella RFC822 era concepito per il trasporto esclusivo di testi, mediante il protocollo SMTP
 - MIME è un'estensione di RFC822 per porre rimedio ad alcuni problemi di SMTP:
 - Incapacità di trasmettere file binari
 - Incapacità di trasmettere caratteri internazionali (solo caratteri a 7 bit)
 - Limite sulla dimensione dei messaggi
 - Problemi sulla formattazione dei testi

MIME & S/MIME

- Specifiche di MIME
 - Nuovi campi di intestazione del messaggio che forniscono informazioni sul body
 - Nuovi formati di contenuti (multimedia e-mail)
 - Codifiche di trasferimento tra i vari formati
- S/MIME
 - Aggiunge specifiche di sicurezza a MIME
 - Supportato in diversi e-mail clients: MS Outlook, Netscape, etc...

Tipi di contenuti di MIME

- Text
- Multipart (il messaggio contiene più parti indipendenti- esempio:multipar/alternative)
- Message
- Image
- Video
- Audio
- Application

Servizi di S/MIME

- enveloped data
 - Contenuto criptato con in allegato la chiave
- signed data
 - Messaggio + hash criptato con chiave privata, il tutto codificato base64
- clear-signed data
 - Messaggio + (hash criptato e codificato base64)
- signed & enveloped data
 - Combinazione di firma e crittografia

Algoritmi crittografici in S/MIME

- Funzioni hash: SHA-1 & MD5
- Firme digitali: DSS & RSA
- Scambio chiave sessione: ElGamal & RSA
- Criptazione dei messaggi: Triple-DES, RC2/40 e altri
- Vi è una procedura per decidere quali algoritmi usare; ogni agente di decisione allega a ogni messaggio una lista, in ordine di preferenza decrescente, degli algoritmi di crittografia che utilizza

Scelta degli algoritmi in S/MIME

1. Scegliere, se possibile, la forma di decriptografia preferita dal destinatario
2. Utilizzare lo stesso algoritmo presente nell'ultimo messaggio ricevuto da parte del destinatario
3. Usare 3DES (si rischia che il destinatario non riesca a fare la decriptazione)
4. Usare RC2/40 (algoritmo meno sicuro)

Se un messaggio va inviato a più destinatari potrebbe essere necessario crittografarlo con più algoritmi!!!

Enveloped Data

1. Generare una chiave di sessione pseudocasuale per un algoritmo di crittografia simmetrico
2. Crittografare la chiave di sessione con la chiave pubblica del destinatario (va fatto per ogni destinatario)
3. Creare, per ogni destinatario, un campo “recipient info” contenente un identificatore del certificato di chiave pubblica utilizzato
4. Crittografare il contenuto del messaggio con la chiave di sessione

Signed Data

1. Selezionare un algoritmo hash
2. Calcolare l'hash del messaggio da firmare
3. Crittografare l'hash con la chiave privata del mittente
4. Preparare il blocco “signer info” contenente il certificato della chiave pubblica del mittente, l’identificatore dell’algoritmo hash utilizzato

Clear-Signed Data

- Il messaggio viene inviato in chiaro
- È quindi visualizzabile da entità non aventi funzionalità S/MIME
- Solo l'hash è criptato e codificato base64

Gestione dei certificati in S/MIME

- S/MIME utilizza certificati X.509 v3
- La gestione è un ibrido tra un meccanismo che fa un uso rigoroso di CA e uno basato sul “web of trust” del PGP
- Ciascun utente S/MIME...
 - Deve generare coppie di chiavi con Diffie-Hellman e DSS; dovrebbe poter generare coppie di chiavi RSA
 - Deve registrare la propria chiave pubblica presso una CA, per poter ricevere un certificato di chiave pubblica X.509
 - Deve poter accedere a un elenco locale di certificati per verificare le firme in ingresso e crittografare i messaggi in uscita

Gestione dei certificati in S/MIME

- Ciascun utente ha quindi una lista di CA di cui si fida
- È un proprio database di coppie di chiavi private/chiavi pubbliche e certificati
- I certificati devono essere firmati dalle CA di cui l'utente si fida

Autorità di certificazione

- Esistono varie aziende che forniscono servizi di certificazione
- Verisign è una delle più note
- Verisign emette diversi tipi di certificati (Digital ID)
- Ciascun Digital ID contiene almeno le seguenti informazioni
 - Chiave pubblica del proprietario
 - Nome e alias del proprietario
 - Data di scadenza
 - Numero di serie del digital ID
 - Nome della CA che ha emesso il certificato
 - Firma digitale della CA che ha emesso il certificato

Verisign: quotazioni Nasdaq



Digital IDs for Secure Email - Digital Signatures from VeriSign, Inc. - Mozilla Firefox

File Modifica Visualizza Vai Segnalibri Strumenti ?

Come iniziare Ultime notizie Mozilla Italia Forum di aiuto

US Home Worldwide Sites Site Map

VeriSign®

Products & Services Solutions Support About VeriSign Existing Customers

You Are Here: US Home > Products & Services > Security Services > Managed PKI Services > PKI Applications > Digital IDs for Secure Email

Digital IDs for Secure Email

In the physical world, you protect your written correspondence by putting it in an envelope before posting. In the online world, sending an email message is like sending a postcard: it is easy to intercept and read as it travels across the Internet. Instead of risking disclosure of your private emails, safeguard them with a VeriSign Digital ID.

Buy Now Digital IDs are available for U.S. \$19.95 and are valid for one year.

Installed in your Web browser or email software, a Digital ID serves as an electronic substitute for sealed envelopes and handwritten signatures, enabling you to:

- ♦ Digitally sign email messages to assure recipients that the email really was sent by you.
- ♦ Encrypt email contents and attachments, protecting them from being read by online intruders. Only your intended recipient can decrypt them.

NOTE: In order to secure your email with a Digital ID, you must use your ID with a POP-based email client.

Certificate Management for Existing Customers

Search

PKI Applications

Other services which are part of the PKI Applications suite include:

Trusted Form Signing
Trusted Messaging
Cable Modem Services
Digital IDs for Secure Instant Messaging

Contact Us Legal Notices Privacy Repository ©1995-2006 VeriSign, Inc. All rights reserved.

Contact Us Please contact support at id-queries@verisign.com.

Special Offers Trial SSL Certificate

Related Resources

Guides Digital ID User Guide
Digital ID Tutorial

Data Sheets Trusted Messaging Solution

News

SanDisk and VeriSign Partner To Extend Consumer Authentication Capabilities to Flash Memory Devices

VeriSign Announces New Service To Protect Against Online Identity Theft

VeriSign to Enhance Strong Authentication Platform with the Introduction of VeriSign® Unified Authentication - Smart Cards



Autorità di certificazione

- Ciascun Digital ID può poi contenere altre informazioni
 - Indirizzo del Certificate Owner
 - Indirizzo di e-mail del Certificate Owner
- Verisign emette tre tipi di certificati con livello crescente di sicurezza...
 - Vi sono quindi certificati di Classe 1, Classe 2 e Classe 3

Applicazioni per l'Autenticazione

Applicazioni per l'autenticazione

- ... ci soffermeremo sulle funzioni che permettono di implementare l'autenticazione, a livello di applicazione, sulle reti internet
- Parleremo di
 - Kerberos
 - Servizio di autenticazione a directory X.509
(servizio utilizzato da S/MIME)

Kerberos

- Servizio di autenticazione sviluppato dal MIT
- Si tratta di un server “fidato” di chiavi
- Kerberos fornisce un server di autenticazione centralizzato la cui funzione è quella di autenticare gli utenti per i server e i server per gli utenti
- Kerberos fa uso soltanto della crittografia simmetrica (DES)
- Ve ne sono due versioni in uso: la 4 e la 5

Kerberos

- Lo scenario di riferimento è quello in cui ci sono delle postazioni accessibili da vari utenti e dei server che forniscono delle risorse
- Un utente si siede presso una postazione e accede tramite la propria password alle risorse del server
- L'obiettivo è quello di rendere questo sistema “sicuro”. Ovvero, solo gli utenti autorizzati devono poter usare le risorse

Requisiti di Kerberos

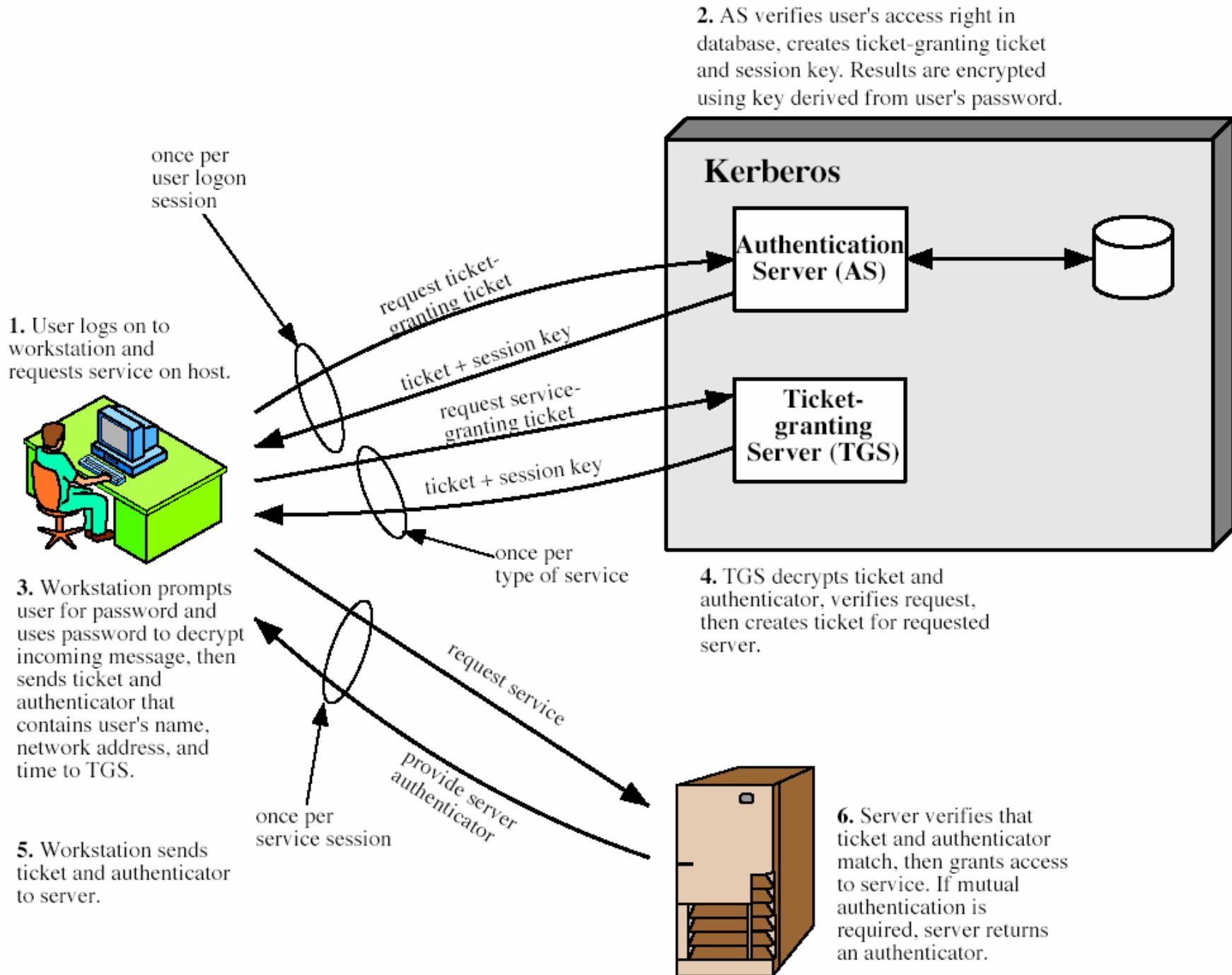
- Il primo documento pubblicato su Kerberos elencava i seguenti requisiti:
 - *Sicurezza*: non deve accadere che un utente sostituisca un altro utente
 - *Affidabilità*: si utilizza un'architettura a server distribuiti
 - *Trasparenza*: teoricamente l'utente non dovrebbe accorgersi che sta avvenendo l'autenticazione, se non fosse per l'immissione di una password
 - *Scalabilità*: Il sistema deve poter supportare un gran numero di client e server
- Kerberos si basa su un protocollo di autenticazione del tipo Needham-Schroeder

Needham-Schroeder Protocol

1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks||ID_A]]$
3. A→B: $E_{Kb}[Ks||ID_A]$
4. B→A: $E_{Ks}[N_2]$
5. A→B: $E_{Ks}[f(N_2)]$

Kerberos 4: Overview

- È uno schema di autenticazione basato su 2 autorità superiori
- Vi è un Authentication Server (AS)
 - Gli utenti inizialmente comunicano con l'AS per identificarsi
 - L'AS fornisce all'utente delle credenziali autenticate non modificabili (ticket granting ticket TGT)
- Vi è un Ticket Granting server (TGS)
 - Gli utenti poi chiedono accesso ai vari servizi al TGS sulla base del TGT



Il Protocollo Kerberos (1/2)

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) AS → C: $E_{K_c}[K_{ctgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{ctgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$ indirizzo di rete di C

(4) TGS → C: $E_{K_{c,tgs}}[K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{ctgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2]$$

$$Ticket_v = E_{K_v}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{tgs}}[ID_C \parallel AD_C \parallel TS_3]$$

$K_{c,tgs}$

Il Protocollo Kerberos (2/2)

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $Ticket_v \parallel Authenticator_c$

(6) V → C: $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$$Ticket_v = E_{K_{c,v}}[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4]$$

$$Authenticator_c = E_{K_{c,v}}[ID_C \parallel AD_C \parallel TS_5]$$

I “reami” Kerberos

- Un ambiente kerberos è fatto di
 - Un server Kerberos (AS + TGS)
 - Un certo numero di clients, tutti registrati presso il server
 - Server di calcolo, ciascuno dei quali condivide una chiave col server Kerberos
- Tale insieme viene detto “reame” (realm)
 - Si tratta tipicamente di un singolo dominio amministrativo
- Se vi sono più reami, allora i server kerberos devono condividere delle chiavi e avere fiducia l'uno negli altri

Kerberos Versione 5

- Sviluppata a metà degli anni '90
- Rispetto alla versione 4 vi sono i seguenti cambiamenti
 - Si possono utilizzare più algoritmi di crittografia (non solo DES)
 - Il protocollo è esteso a reti che non hanno indirizzi in formato IP
 - I ticket possono avere durata arbitraria e non più una durata massima di $5 \times 256 = 1280$ minuti
 - Possibilità di inoltro automatico dell'autenticazione

Kerberos Versione 5

- Autenticazione tra N realm diversi non richiede N^2 relazioni tra i server Kerberos
- Viene eliminata la doppia crittografia presente nei messaggi 2 e 4
- Kerberos versione 5 è pubblicato come Internet standard nella RFC 1510

The screenshot shows a Mozilla Firefox window with the title bar "Kerberos: The Network Authentication Protocol - Mozilla Firefox". The menu bar includes "File", "Modifica", "Visualizza", "Vai", "Segnalibri", "Strumenti", and "?". The toolbar includes standard icons for back, forward, stop, and home, along with a search field containing "http://web.mit.edu/kerberos/" and a refresh button. Below the toolbar, there are links for "Come iniziare", "Ultime notizie", "Mozilla Italia", and "Forum di aiuto".

Kerberos: The Network Authentication Protocol



- [What is Kerberos?](#)
- [Security Advisories](#)
- Kerberos Releases
 - Current release: [krb5-1.4.3](#)
 - [Historical releases of MIT krb5](#)
- Download
 - [Sources and binaries from MIT](#)
- Releases in testing
 - [The krb5-current Snapshots](#) (for developers only)
- Documentation
 - [Documentation for the krb5-1.4.3 release](#)
 - The [comp.protocols.kerberos](#) FAQ (at NRL; maintained by Ken Hornstein)
 - [How do the new US export regulations affect Kerberos?](#)
 - [Papers about the Kerberos protocol](#)
 - [Kerberos Y2K statement](#)
- [The MIT Kerberos Team](#)
- [Contact Information](#)
- Other Resources
 - [Mailing lists](#)
 - [comp.protocols.kerberos](#) newsgroup
 - [USC/ISI Kerberos Page](#)
 - [Oak Ridge National Laboratory's "How to Kerberize your Site"](#)

Recent News

Servizio di Autenticazione X.509

- La raccomandazione ITU-T X.509 fa parte delle raccomandazioni X.500 che definiscono un servizio a directory
- La directory è un server (o un insieme di server) che gestisce un database di informazioni sugli utenti
- X.509 definisce una struttura per fornire servizi di autenticazione degli utenti: La directory può contenere i certificati di autenticazione degli utenti
- I certificati e i protocolli X.509 sono ampiamente utilizzati (S/MIME, IPSec, SSL/TLS, SET)

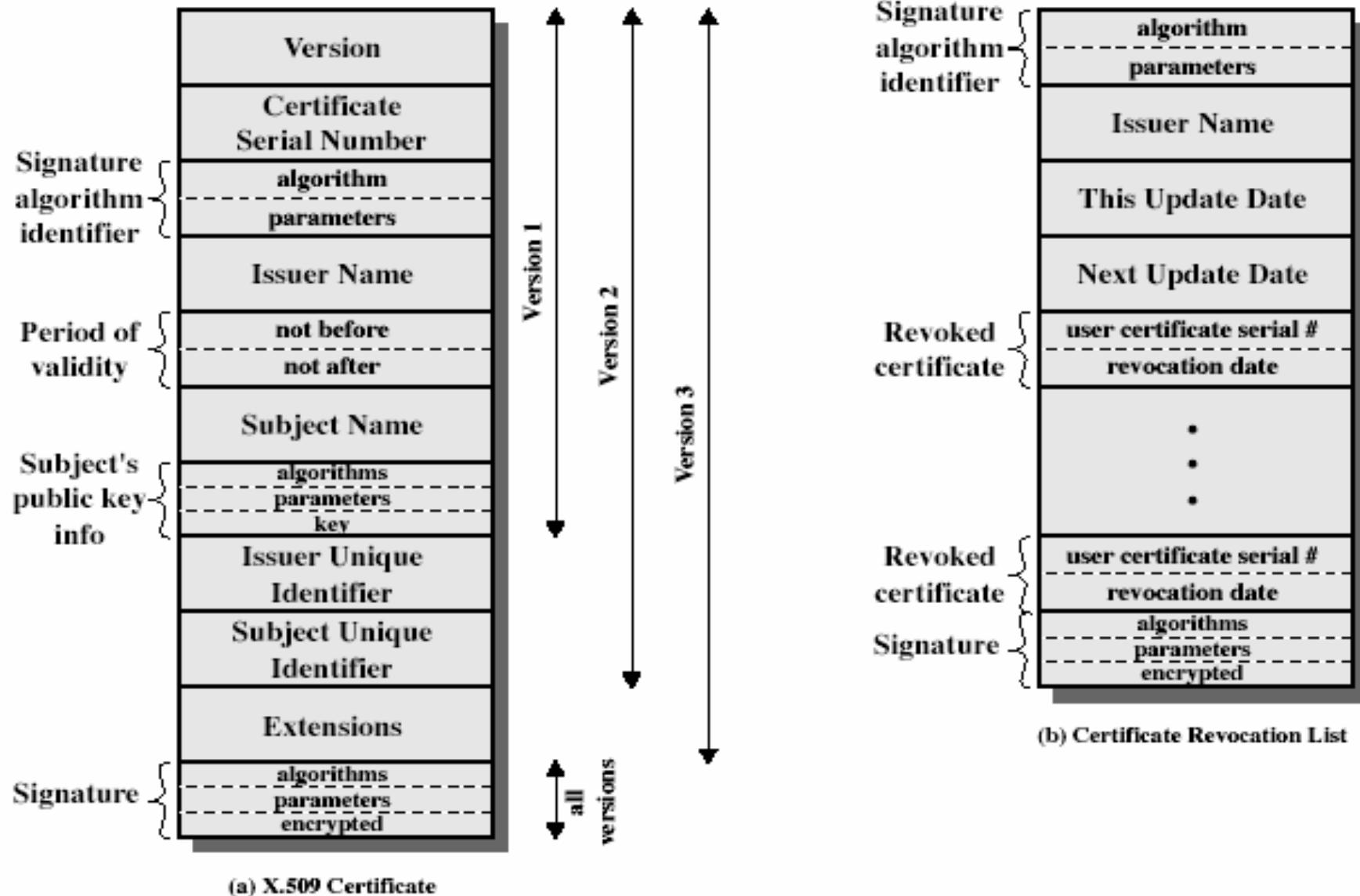
Servizio di Autenticazione X.509

- X.509 è stato emesso inizialmente nel 1988; è stato poi rivisto nel 1993 e nel 1995 (versione 3)
- X.509 si basa sull'uso della crittografia a chiave pubblica e delle firme digitali
 - Lo standard non è agganciato a qualche particolare algoritmo, anche se viene raccomandato RSA
- Il cuore di X.509 è il certificato utente, che è creato da una CA e inserito nella directory. Il server che ospita i certificati non coincide quindi in generale con la CA

Certificati X.509

- Un certificato X.509 contiene...
 - versione (1, 2, or 3)
 - serial number (unique within CA) identifying certificate
 - signature algorithm identifier
 - issuer X.500 name (CA)
 - period of validity (from - to dates)
 - subject X.500 name (name of owner)
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)
 - signature (of hash of all fields in certificate)
- CA<<A>> indica il certificato di A firmato da CA

Certificati X.509



Come si ottiene un Certificato?

- Ogni utente che conosce la chiave pubblica della CA può leggere i certificati emessi da questa CA
- Solo la CA può modificare i certificati
- Poichè i certificati non possono essere nè alterati nè falsificati, possono essere posti in una directory pubblica

Gerarchia delle CA

- Come possono autenticarsi due utenti che fanno riferimento a due diverse CA?

Esempio: A ha un certif. firmato dalla CA X1, e B da X2

A ottiene il certificato di X2 firmato da X1; può quindi essere certo della chiave pubblica di X2 e quindi leggere il certificato di B:

X1<<X2>>X2<>

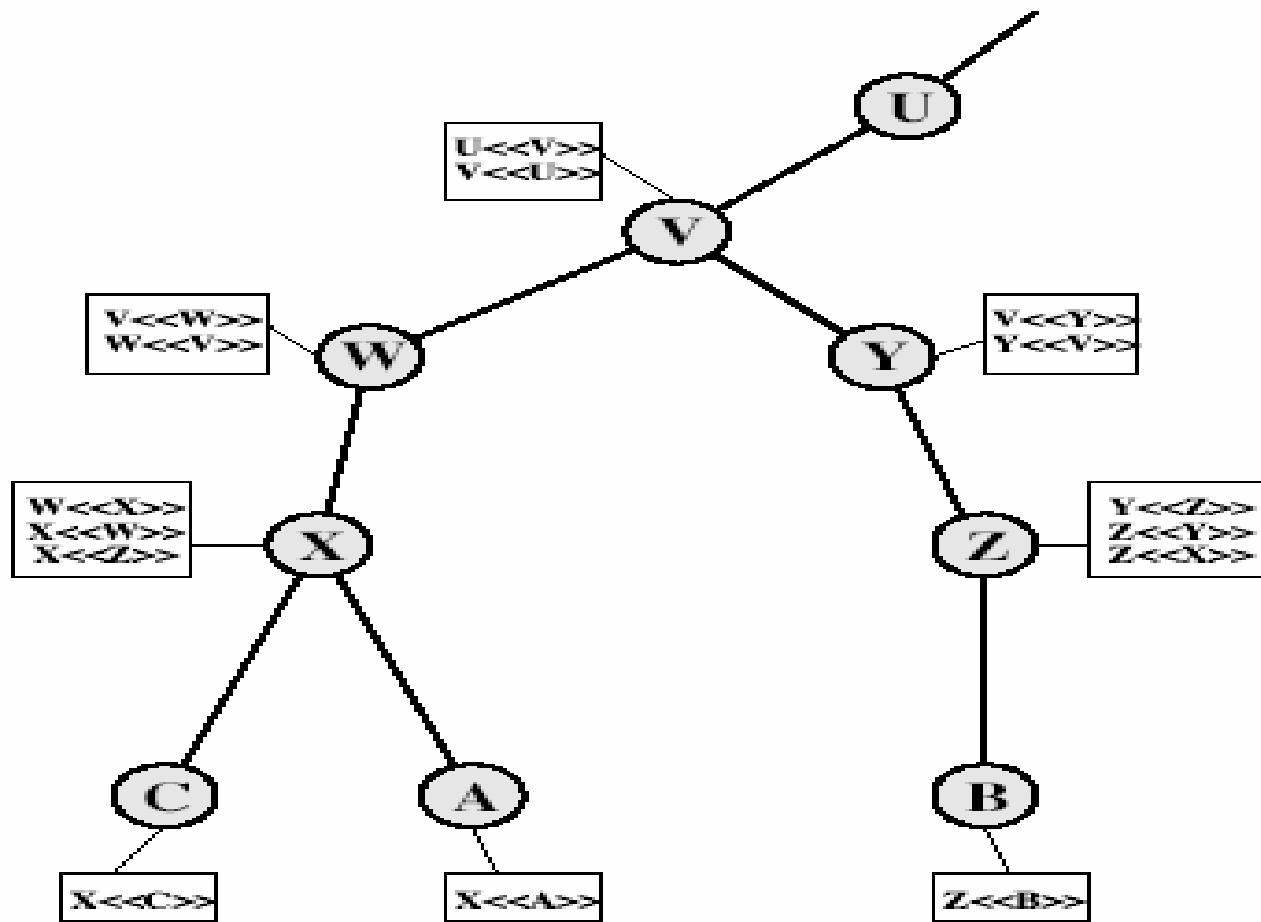
Analogamente, B ottiene la chiave pubblica di A con la catena inversa

X2<<X1>>X1<<A>>

Gerarchia delle CA

- In generale, le varie CA sono disposte in modo gerarchico
- Ciascuna CA ha i certificati delle CA antecedenti e susseguenti nella gerarchia
- Un utente può quindi verificare i certificati emessi da qualsiasi CA che fa parte della gerarchia

Esempio di gerarchia di CA



Revoca dei certificati

- I certificati hanno un periodo di validità
- Possono essere revocati in anticipo se:
 1. La chiave privata è violata
 2. L'utente non è più certificato da una data CA
 3. Il certificato è stato violato
- Ciascuna CA ha una lista dei certificati revocati, la Certificate Revocation List (CRL)
- Gli utenti devono controllare che i certificati non siano inseriti in tale elenco

Procedure di Autenticazione

- X.509 prevede tre diverse procedure di autenticazione
- One-Way Authentication (es. e-mail)
- Two-Way Authentication (sessioni interattive con timestamp)
- Three-Way Authentication (sessioni interattive senza timestamp)
- Tutte usano firme a chiave pubblica

One-Way Authentication

- 1 messaggio (A->B) utilizzato per
 - Provare l'identità di A e che il messaggio è stato trasmesso da A
 - Provare che il messaggio era indirizzato a B
 - Stabilire l'integrità e l'originalità del messaggio
- Il messaggio include un timestamp, un nonce, l'identità di B, e una **chiave di sessione** (criptata con la chiave pubblica di B) ed è firmato da A

Two-Way Authentication

- 2 messaggi ($A \rightarrow B$, $B \rightarrow A$) che in aggiunta ai precedenti
 - Stabiliscono l'identità di B ed il fatto che la risposta proviene da B
 - Provano che la risposta è destinata a A
 - Provano l'integrità e l'originalità della risposta
- La risposta contiene il nonce originale da A, un timestamp, la chiave di sessione criptata con la chiave pubblica di A, e un nonce di B

Three-Way Authentication

- 3 messaggi ($A \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$) che permettono di effettuare l'autenticazione bidirezionale senza far uso dei timestamp, ovvero senza dover sincronizzare gli orologi
- In aggiunta ai messaggi precedenti, vi è la ritrasmissione, da A a B, del nonce generato da B e firmato da A
- Questo permette di non far affidamento sui timestamp e prevenire gli attacchi a replay

IP Security

Cenni

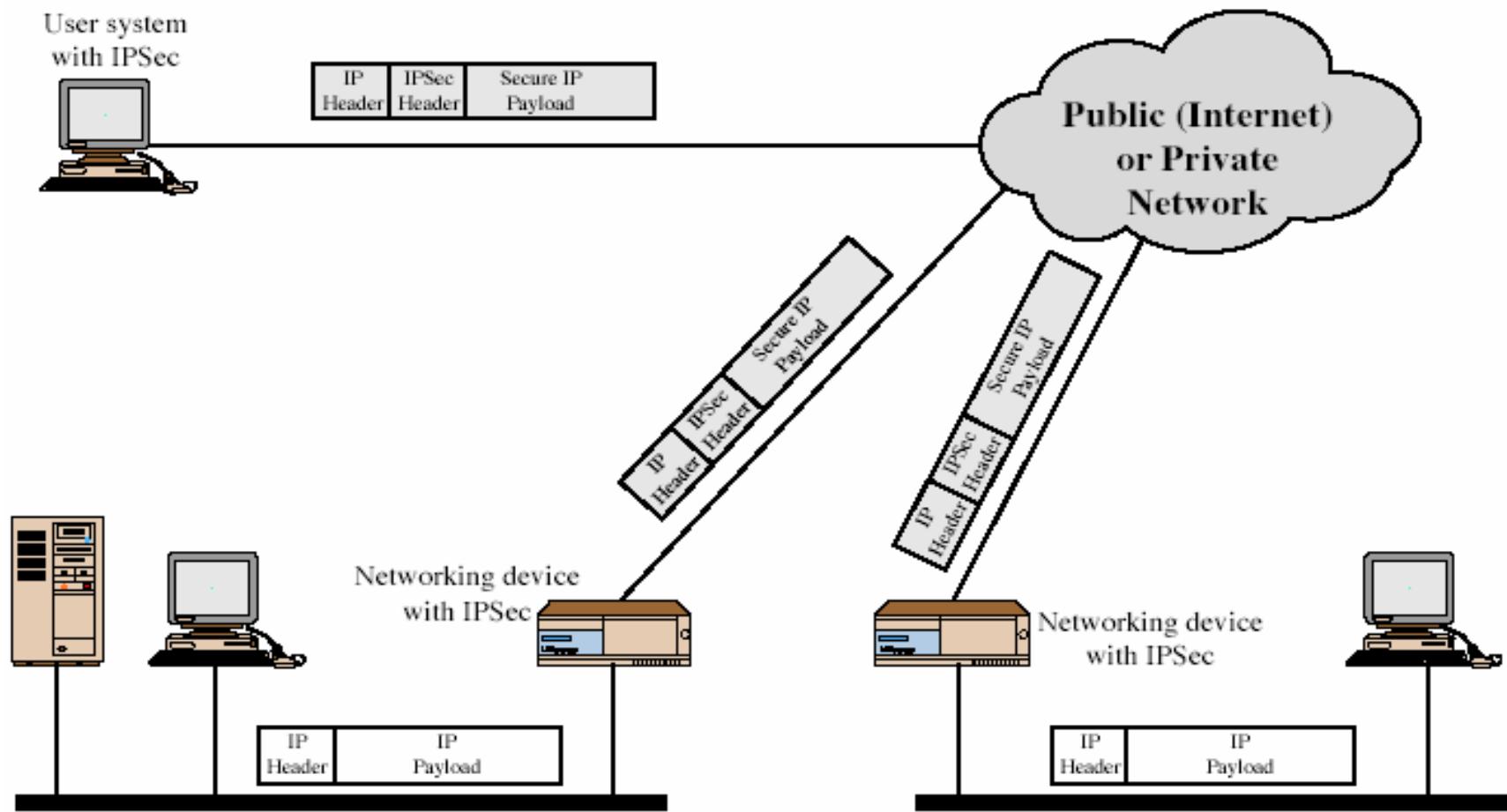
IP Security

- Sinora abbiamo considerato alcune applicazioni specifiche che implementano la sicurezza
 - eg. S/MIME, PGP, Kerberos
- Sarebbe preferibile tuttavia avere un meccanismo che permetta connessioni sicure su una rete non sicura per tutte le applicazioni

IPSec

- IPSec fornisce la sicurezza a livello IP
- IPSec fornisce
 - autenticazione
 - segretezza
 - Un meccanismo di gestione delle chiavi
- Si può utilizzare su reti LAN, reti WAN e, ovviamente, sulla rete Internet

Esempio di uso di IPSec



Vantaggi e Caratteristiche di IPSec

- Se implementato in un firewall o router, IPSec fornisce un livello di sicurezza che riguarda tutto il traffico che attraversa il perimetro. Il firewall si occupa del sovraccarico computazionale legato alla sicurezza
- IPSec è sotto il livello di trasporto, quindi è trasparente alle applicazioni superiori
- IPSec può essere trasparente agli utenti finali
- IPSec può garantire sicurezza anche a singoli utenti (vedi creazione di una VPN di un utente in trasferta con la propria LAN aziendale)

Architettura IPsec

- Le specifiche di IPsec sono alquanto complesse
- Sono definite in diverse RFC...
 - Ad es. RFC 2401/2402/2406/2408
- L'uso di IPsec è obbligatorio in IPv6
- Algoritmi crittografici utilizzati:
 - 3DES a tre chiavi, RC5, IDEA, 3IDEA a tre chiavi, CAST, Blowfish

Web Security

Web Security

- Il World Wide Web è ormai un elemento fondamentale della vita comune
- Internet & Web sono in realtà vulnerabili
- Vi sono una serie di minacce
 - integrità
 - confidenzialità
 - denial of service
 - autenticazione
- Vi è bisogno quindi di meccanismi supplementari di sicurezza

Web Security

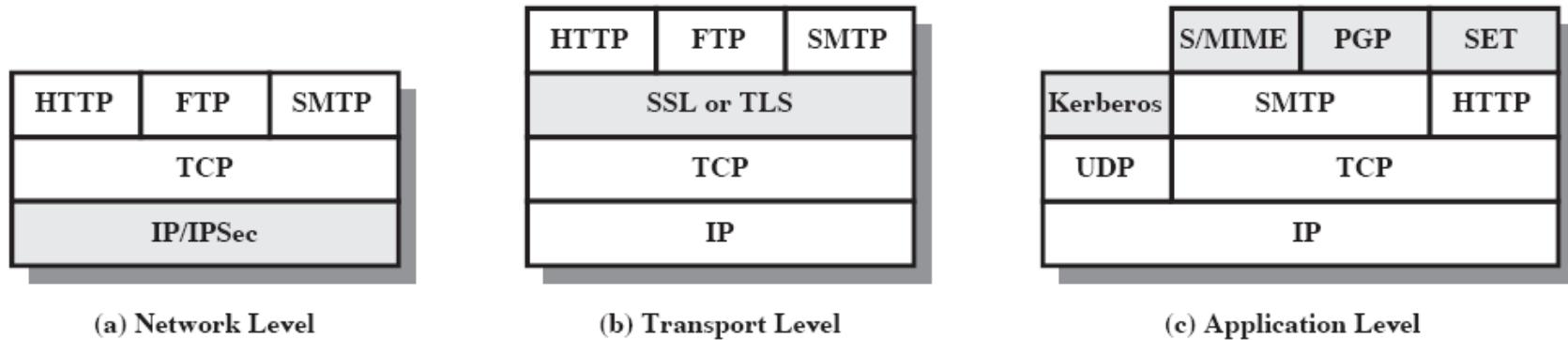
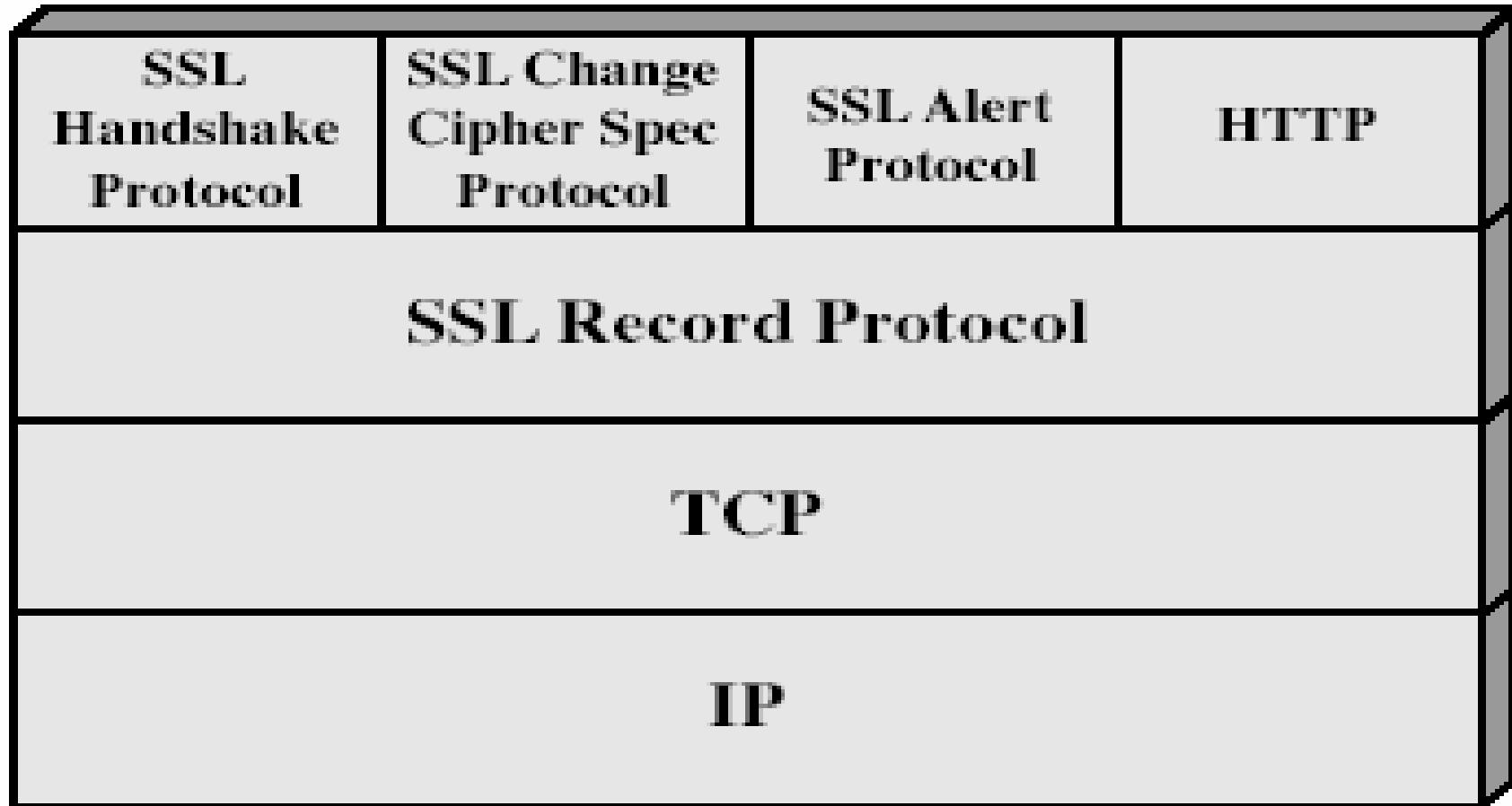


Figure 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

SSL (Secure Socket Layer)

- È un servizio di sicurezza a livello del protocollo di trasporto
- Sviluppato originariamente da Netscape
- La versione 3 è stata sviluppata con revisione pubblica
- Tale versione è diventata uno standard Internet col nome di TLS (Transport Layer Security)
- SSL si poggia su TCP per ottenere un servizio di trasmissione punto-punto affidabile
- SSL ha due livelli protocollari

Architettura SSL



SSL (Secure Socket Layer)

- Il protocollo SSL Record Protocol fornisce i servizi di sicurezza di base per i vari protocolli di livello superiore, come ad esempio http, che può operare su SSL
- Nell'ambito di SSL sono definiti tre protocolli di alto livello necessari alla gestine degli scambi SSL

Architettura SSL

- **Sessione SSL**
 - È un'associazione tra client e server
 - È creata dal protocollo di handshake
 - Definisce un insieme di parametri di sicurezza crittografica che possono essere usati da più connessioni
 - Viene quindi evitata la negoziazione per ogni connessione
- **Connessione SSL**
 - Un link di comunicazione punto-punto e temporaneo
 - Ciascuna connessione può essere associata ad un'unica sessione SSL

SSL Record Protocol

È un protocollo di trasporto che fornisce i servizi di

- **confidenzialità**

- Tramite crittografia simmetrica con chiave scambiata tramite il protocollo di handshake
- IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
- Il messaggio è compresso prima della crittografia

- **integrità**

- Usa un MAC con chiave segreta condivisa
- Si tratta di una variante di HMAC

SSL Record Protocol

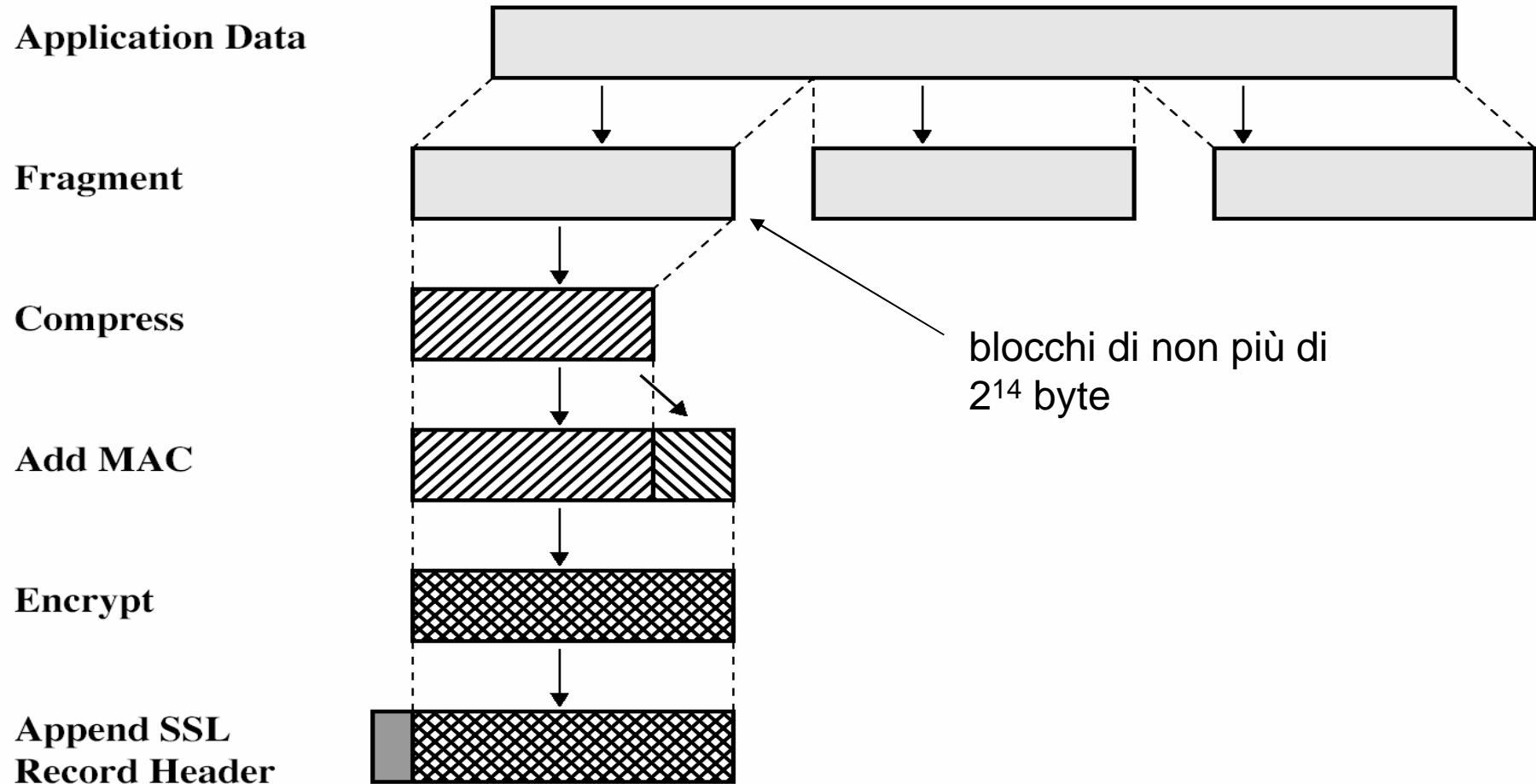


Figure 17.3 SSL Record Protocol Operation

SSL Change Cipher Spec Protocol

- È uno dei 3 protocolli specifici SSL che usano lo SSL Record protocol
- Si tratta di un singolo messaggio
- Lo scopo di tale messaggio è fare in modo che lo stato provvisorio venga copiato nello stato corrente, aggiornando la cifratura che verrà usata nella sessione corrente

SSL Alert Protocol

- Trasporta messaggi di allarme relativi alla connessione SSL
- I messaggi sono caratterizzati da due diversi gradi di importanza:
 - warning or fatal
- Allarmi specifici sono i seguenti:
 - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
 - close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- I messaggi sono compressi e criptati come tutti i dati SSL

SSL Handshake Protocol

- Permette al client e al server di:
 - Autenticarsi a vicenda
 - Negoziare gli algoritmi MAC e crittografici da utilizzare
 - Negoziare le chiavi da usare
- Prevede lo scambio di una serie di messaggi in varie fasi
 - Stabilisce le capacità crittografiche del client e del server
 - Provvede all'autenticazione reciproca del client e del server e allo scambio delle chiavi

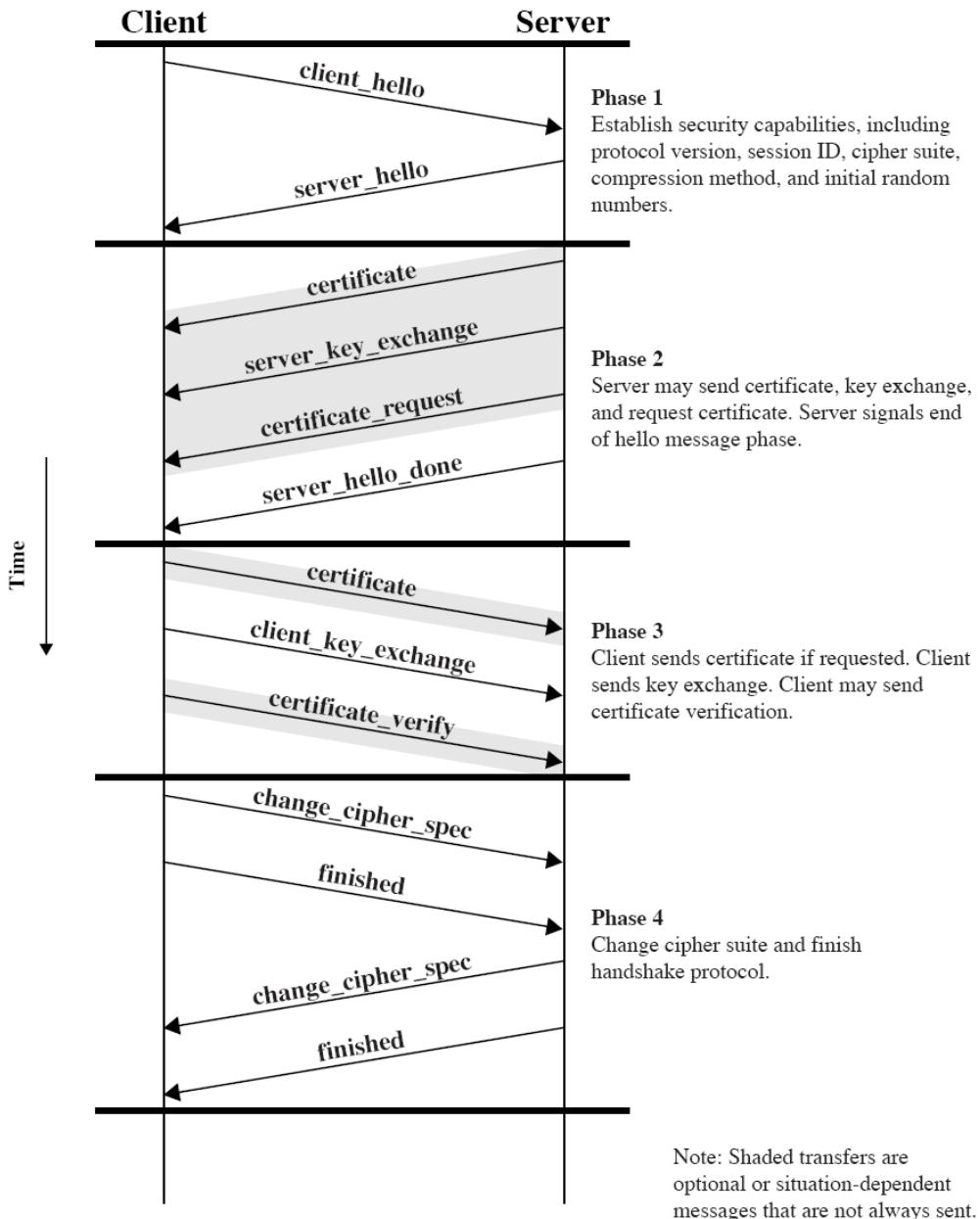


Figure 17.6 Handshake Protocol Action

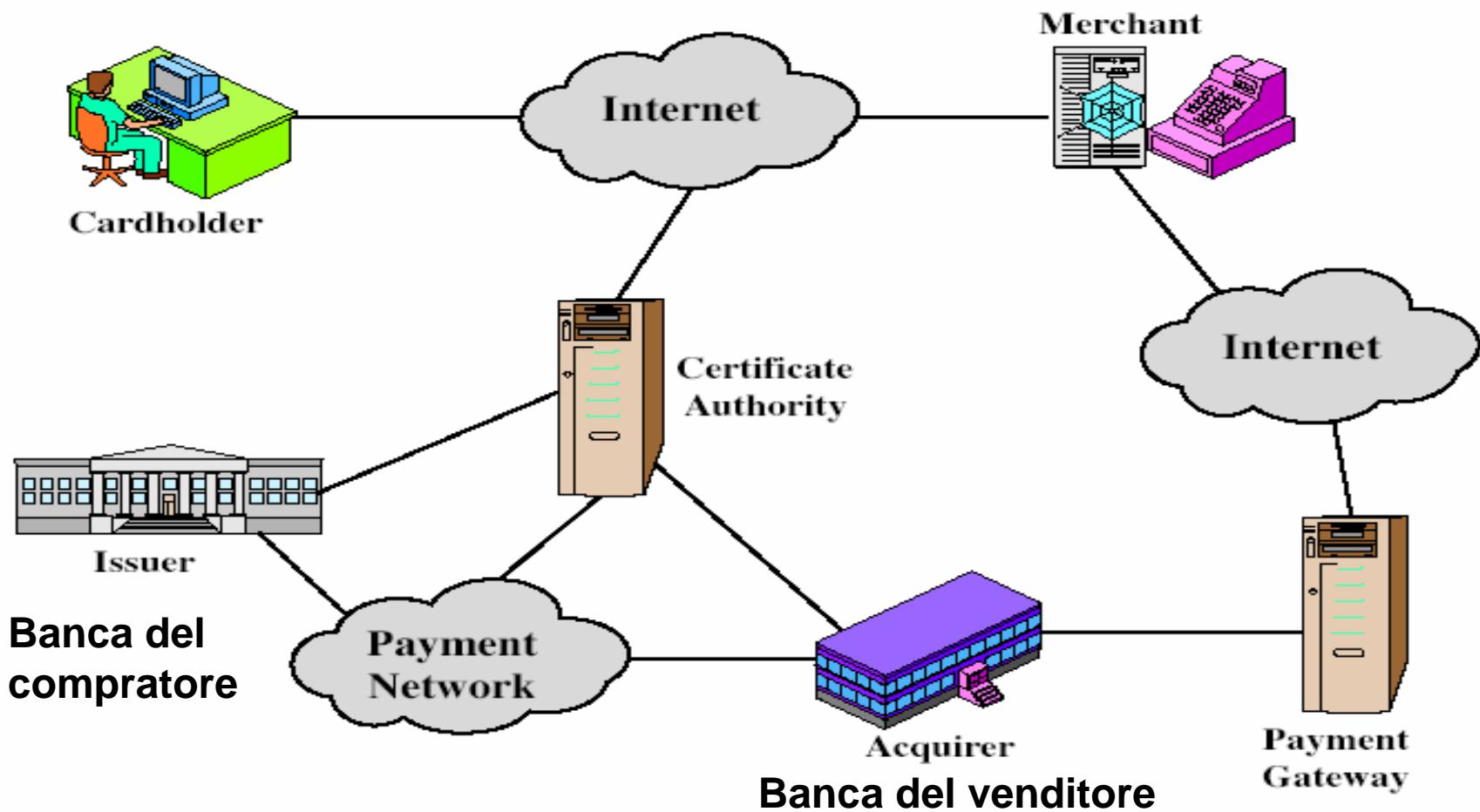
SET

Secure Electronic Transactions

Secure Electronic Transactions (SET)

- È una specifica di sicurezza espressamente progettata per proteggere l'uso della carta di credito su Internet
- Fu sviluppato nel 1996 su richiesta di Visa e Mastercard
- Non è un sistema di pagamento...
- ... ma una suite di protocolli che forniscono
 - Comunicazioni sicure tra i soggetti coinvolti
 - Autenticazione tramite l'uso di certificati X.509v3
 - Privacy confinando l'informazione solo a chi ne ha bisogno effettivamente

Componenti di SET



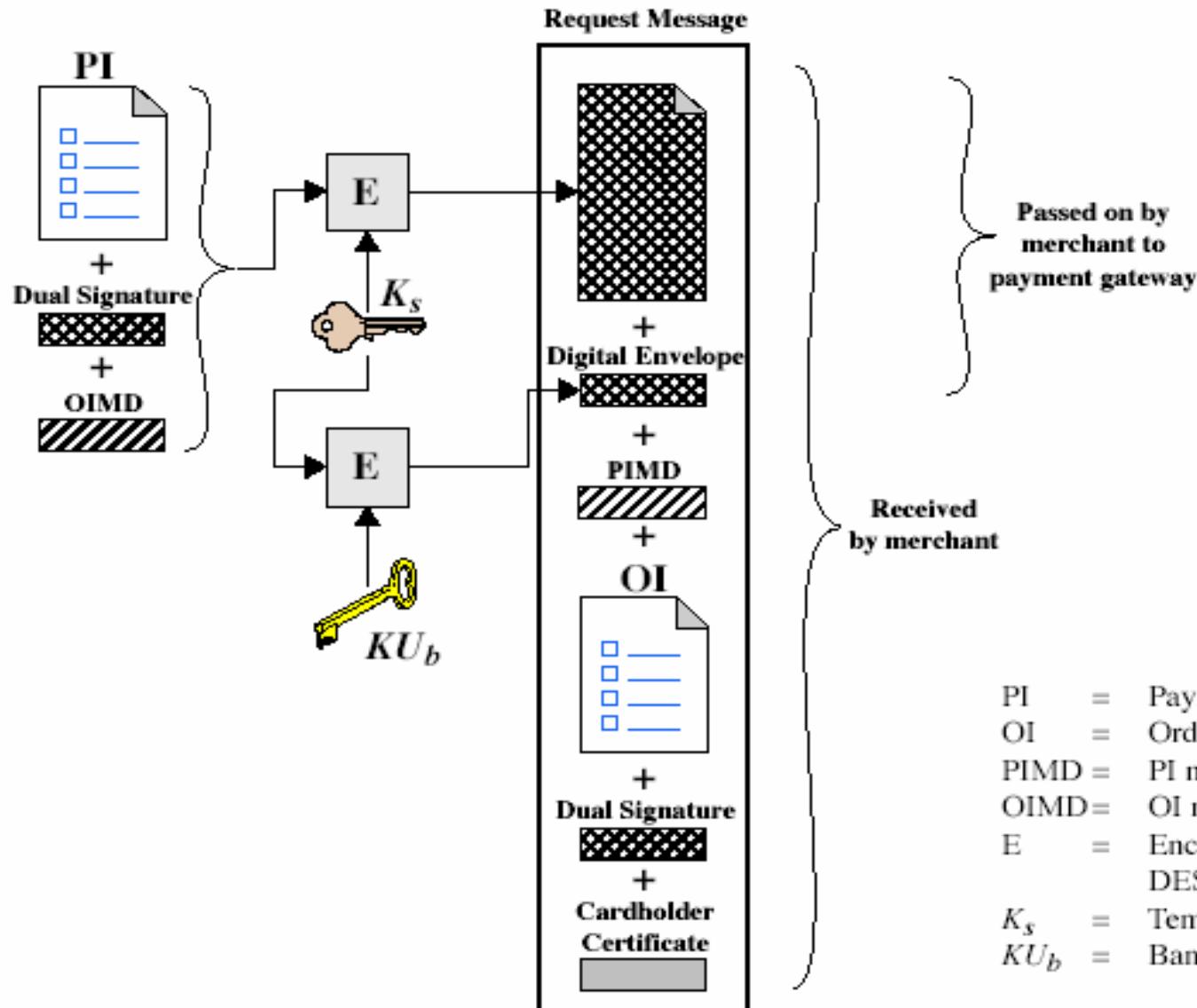
Una Transazione SET

1. Il cliente apre un conto
2. Il cliente riceve un proprio certificato
3. I commercianti hanno anch'essi i loro certificati
4. Il cliente emette un ordine
5. Il commerciante è “verificato”
6. L'ordine e il pagamento vengono inviati
7. Il commerciante richiede l'autorizzazione al pagamento
8. Il commerciante conferma l'ordine
9. Il Commerciano fornisce il prodotto richiesto
10. Il Commerciano richiede il pagamento

Firma Doppia

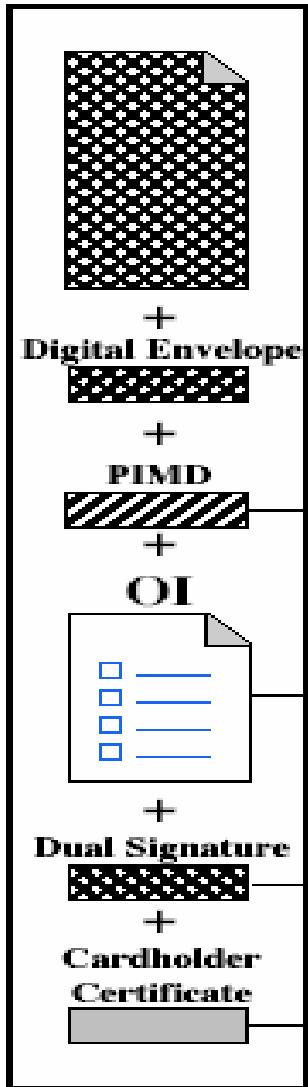
- Il cliente crea due messaggi
 - order information (OI) per il commerciante
 - payment information (PI) per la banca
- Le due informazioni devono essere distinte (la banca non ha bisogno di sapere la OI)
- Tuttavia devono poter essere collegate per risolvere le dispute
- Si usa quindi una doppia firma
 - Viene firmata la concatenazione degli hash dell'OI e della PI

Richiesta di acquisto - Cliente



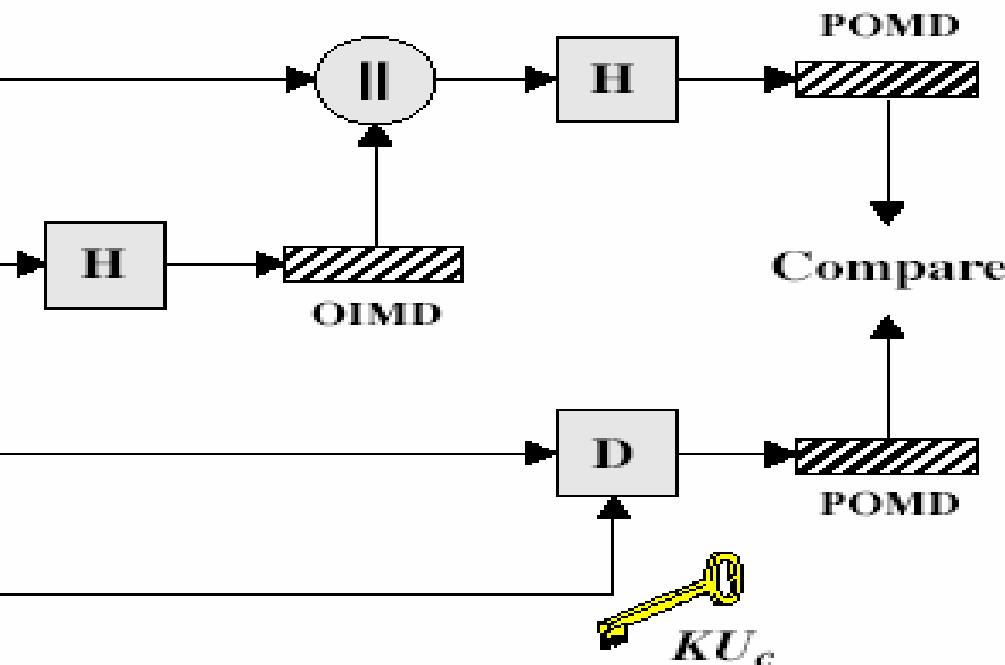
Richiesta di Acquisto - Commercianti

Request Message



Passed on by merchant to payment gateway

OI	= Order Information
OIMD	= OI message digest
POMD	= Payment Order message digest
D	= Decryption (RSA)
H	= Hash function (SHA-1)
KU_c	= Customer's public signature key



Richiesta di Acquisto - Commercianti

1. Verifica il certificato dell'acquirente
2. Verifica la doppia firma usando la chiave pubblica dell'acquirente per essere sicuri che non sia stato alterato e che sia stato effettivamente inviato dall'acquirente
3. Elabora l'ordine e invia la PI al payment gateway per l'autorizzazione
4. Invia una risposta all'acquirente

Autorizzazione del payment gateway

1. Verifica tutti i certificati
2. Deccripta la busta digitale per ottenere la chiave segreta e deccripta il messaggio inviato dal commerciante
3. Verifica la firma del commerciante
4. Verifica la doppia firma
5. Verifica che l'identificativo di transizione ricevuto dal venditore coincide con quello inviato dall'acquirente
6. Richiede e riceve un'autorizzazione dall'issuer (banca)
7. Invia l'autorizzazione al commerciante

Cattura del Pagamento

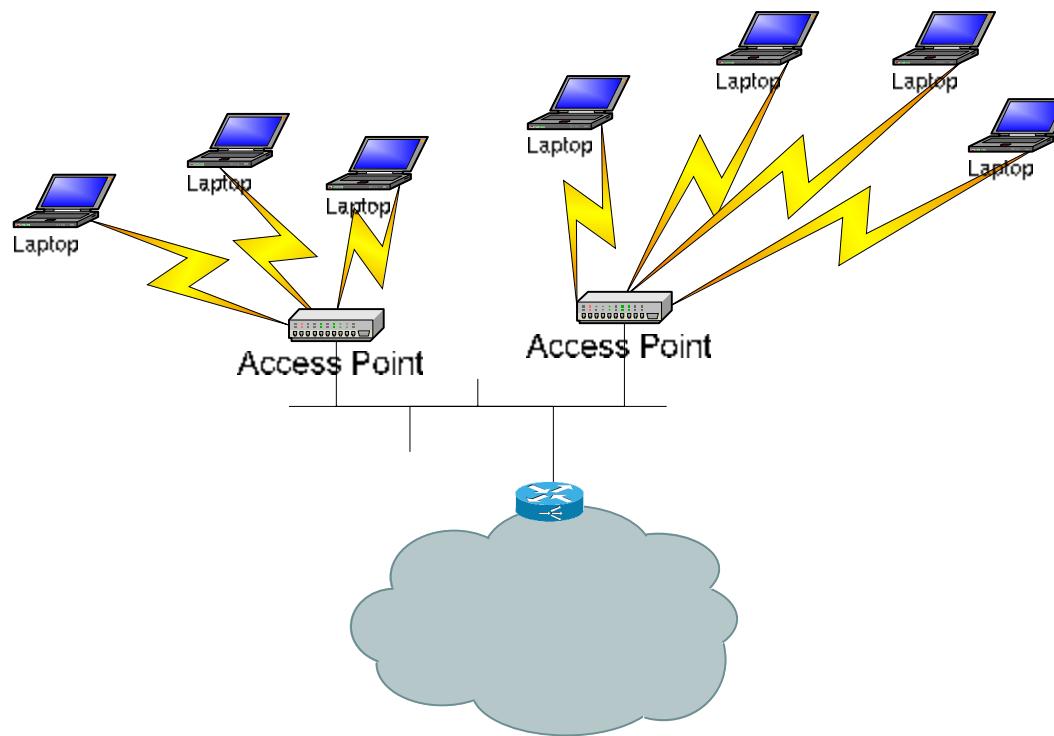
- Il commerciante invia al payment gateway una richiesta di cattura del pagamento
- Il gateway controlla la richiesta
- Poi dispone il trasferimento dei fondi sul conto del venditore
- Informa poi il commerciante dell'avvenuto pagamento

Crittografia e Reti Wi-fi

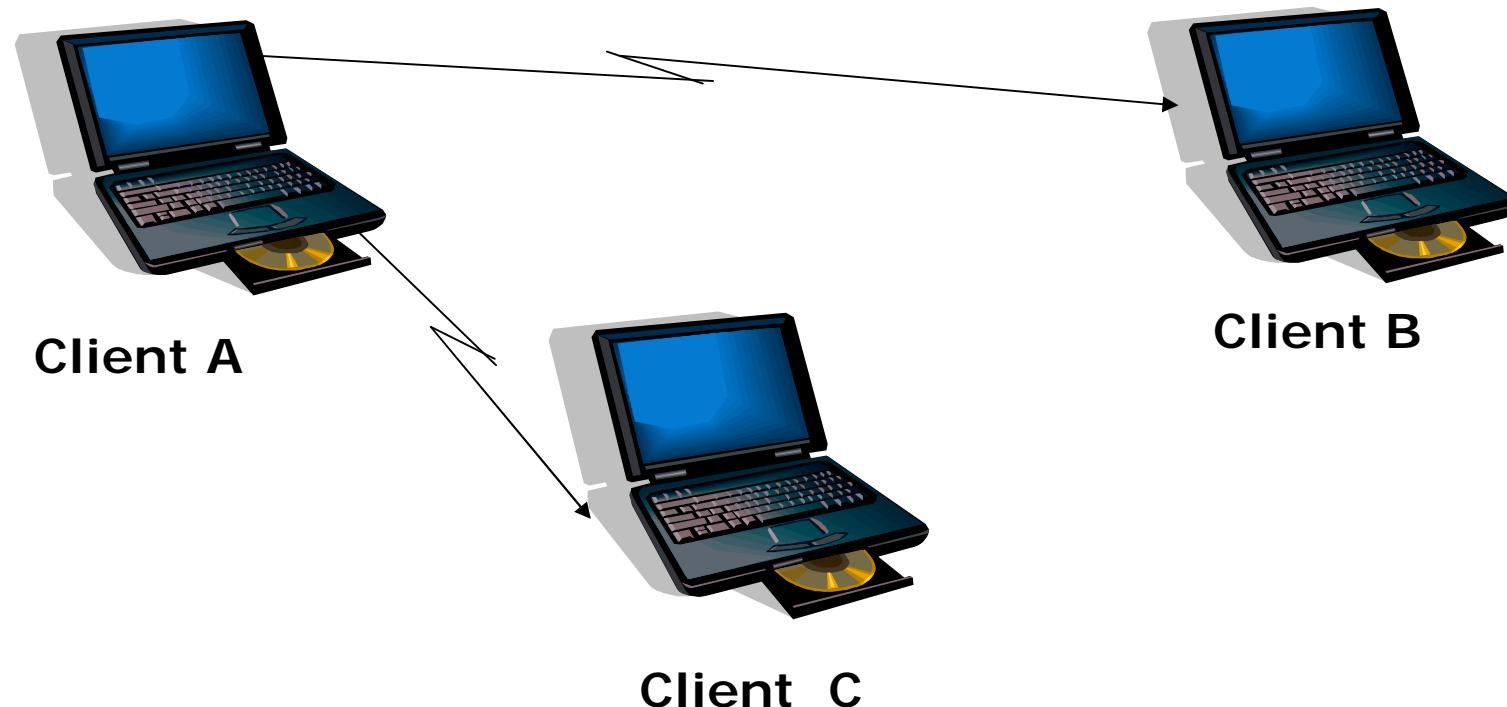
Reti Wi-Fi

- Ci focalizziamo sulle reti IEEE802.11
- Modalità operative
 - Ad Hoc mode (Independent Basic Service Set - IBSS)
 - Infrastructure mode (Basic Service Set - BSS)

802.11 Infrastructure Mode



Ad-Hoc mode



Laptop users wishing to share files could set up an ad-hoc network and share files without need for external media.

Sicurezza nelle WLAN

- La trasmissione sul canale wireless è ovviamente meno sicura di quella su un canale wired
- Meccanismi di sicurezza:
 - SSID broadcast
 - MAC filtering
 - WEP protocol
- **I meccanismi di sicurezza sono disabilitati di default!!!**

Service Set Identifier (SSID)

- Dovrebbe limitare gli accessi identificando l'area coperta dall'access point
- L'AP periodicamente invia il proprio SSID
- I terminali hanno la possibilità di associarsi ad un AP sulla base del suo SSID
- In questo modo sono facilmente note le reti presenti in una certa zona...
- Esistono tool di analisi (es. AirMagnet, Netstumbler, AiroPeek) per l'identificazione dell' SSID.
- Alcuni produttori usano dei nomi di default ben noti (es. CISCO usa tsunami)
- **... in realtà l'SSID non è un dispositivo di sicurezza!!**

MAC address filtering

Esempio di MAC address: 00-12-F0-62-3D-1A

Vantaggi:

- consente maggiore sicurezza

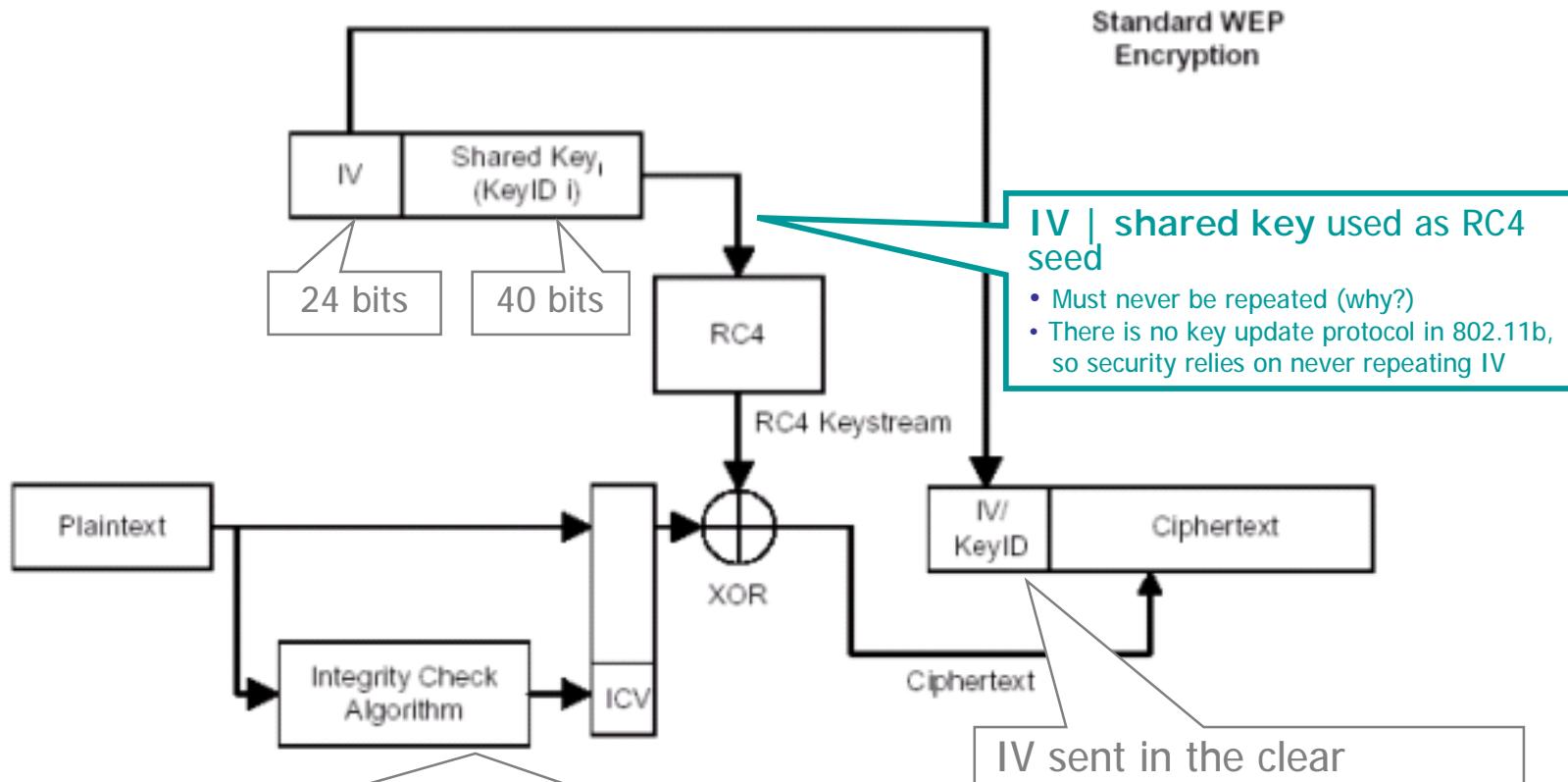
Svantaggi:

- aumenta la complessità
- riduce la scalabilità
- hackers esperti possono comunque aggirare tale sistema

Wired Equivalent Privacy

- Designed to provide confidentiality to a wireless network similar to that of standard LANs.
- WEP is essentially the RC4 symmetric key cryptographic algorithm (same key for encrypting and decrypting).
- Transmitting station concatenates 40 bit key with a 24 bit Initialization Vector (IV) to produce pseudorandom key stream.
- Plaintext is XORed with the pseudorandom key stream to produce ciphertext.
- Ciphertext is concatenated with IV and transmitted over the Wireless Medium.
- Receiving station reads the IV, concatenates it with the secret key to produce local copy of the pseudorandom key stream.
- Received ciphertext is XORed with the key stream generated to get back the plaintext.

WEP: Integrità e Confidenzialità



CRC-32 checksum is linear in \oplus : if attacker flips some bit in plaintext, there is a known, plaintext-independent set of CRC bits that, if flipped, will produce the same checksum

no integrity!

RC4

- Progettato nel 1987 da Ron Rivest
- Fu conservato come segreto industriale e reso noto nel 1994 da un hacker
- Chiave a lunghezza variabile tra 1 e 256 bytes
- Si utilizza un vettore di stato S fatto di 256 byte

RC4

```
for i=0:255,  
    S(i)=i  
    T(i) = K (i mod keylength)  
end;
```

```
j=0;  
for j=0:255,  
    j=(j+ S(i) + T(i)) mod 256  
    swap(S(i),S(j))  
end  
% S contiene ora tutti i numeri tra 0 e 255
```

RC4

i=j=0

while (testo da cifrare presente)

i=(i+1) mod 256

j=(j+S(i)) mod 256

Swap(S(i),S(j))

t= (S(i) +S(j)) mod 256

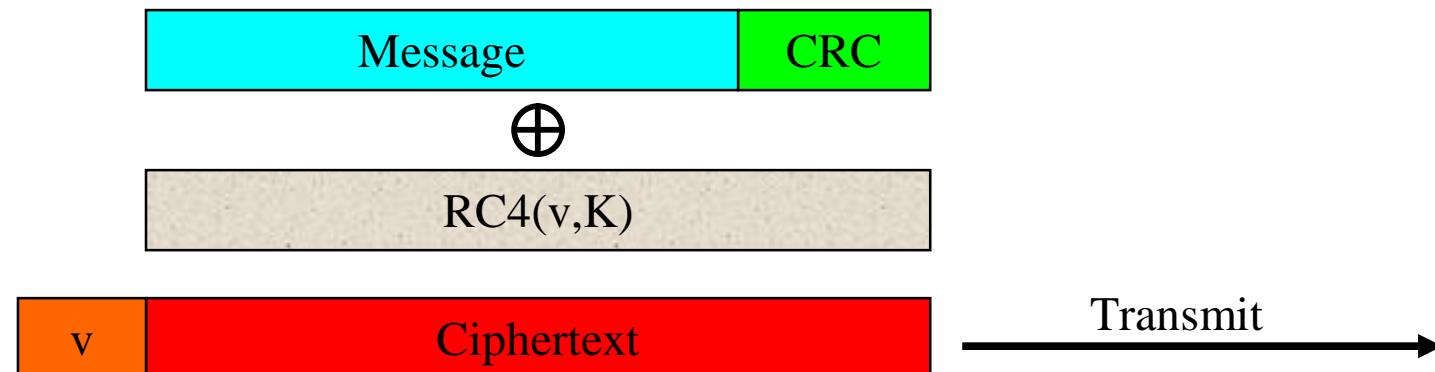
k=S(t)

end

k va poi sommato al byte di testo in chiaro

WEP Overview

- WEP relies on a shared key K between communicating parties
- **Checksum:** For a message M, we calculate c(M). The plaintext is P={M,c(M)}
- **Encryption:** The plaintext is encrypted using RC4. RC4 requires an initialization vector (IV) v, and the key K. Output is a stream of bits called the keystream. Encryption is XOR with P.
$$C = P \oplus \text{RC4}(v, K)$$
- **Transmission:** The IV and the ciphertext C are transmitted.



Obiettivi di sicurezza di WEP

- WEP had three main security goals:
 - Confidentiality: Prevent eavesdropping
 - Access Control: Prevent inappropriate use of 802.11 network, such as facilitate dropping of not-authorized packets
 - Data Integrity: Ensure that messages are not altered or tampered with in transit
- The basic WEP standard uses a 40-bit key (with 24bit IV)
- Additionally, many implementations allow for 104-bit key (with 24bit IV)
- None of the three goals are provided in WEP due to serious security design flaws and the fact that it is easy to eavesdrop on WLAN

Riuso della chiave in WEP

- Vernam-style stream ciphers are susceptible to attacks when same IV and key are reused:

$$C_1 = P_1 \oplus \text{RC4}(v, K)$$

$$C_2 = P_2 \oplus \text{RC4}(v, K)$$

$$\begin{aligned} C_1 \oplus C_2 &= P_1 \oplus \text{RC4}(v, K) \oplus P_2 \oplus \text{RC4}(v, K) \\ &= P_1 \oplus P_2 \end{aligned}$$

- Particularly weak to known plaintext attack: If P_1 is known, then P_2 is easy to find (as is RC4).
 - This might occur when contextual information gives P_1 (e.g. application-level or network-level information reveals information)
- Even so, there are techniques to recover P_1 and P_2 when just $(P_1 \text{ XOR } P_2)$ is known
 - Example, look for two texts that XOR to same value

(In)Sicurezza del WEP

- WEP's engineers were aware (it seems??) of this weakness and required a per-packet IV strategy to vary key stream generation
- Problems:
 - Keys, K, typically stay fixed and so eventual reuse of IV means eventual repetition of keystream!!
 - IVs are transmitted in the clear, so its trivial to detect IV reuse
 - Many cards set IV to 0 at startup and increment IV sequentially from there
 - Even so, the IV is only 24 bits!
- Calculation: Suppose you send 1500 byte packets at 5Mbps, then 2^{24} possible IVs will be used up in 11.2 hours! (5 hours at 11Mbps)
- Even worse: we should expect to see at least one collision after 5000 packets are sent! (due to birthday paradox)
- Thus, we will see the same IV again... and again...

Recupero delle chiavi

- Get access point to encrypt a known plaintext
 - Get victim to send an email with known content
- If attacker knows plaintext, it is easy to recover keystream from ciphertext
 - $C \oplus M = (M \oplus \text{RC4}(IV, key)) \oplus M = \text{RC4}(IV, key)$
 - Not a problem if this keystream is not re-used
- Even if attacker doesn't know plaintext, he can exploit regularities (plaintexts are not random)
 - For example, IP packet structure is very regular

Recupero delle chiavi

- Misuse of RC4 in WEP is a design flaw with no fix
 - Longer keys do not help!
 - The problem is re-use of IVs, their size is fixed (24 bits)
 - Attacks are passive and very difficult to detect

Da non fare!!!!

Ingredients: Laptop (with 802.11b card, GPS, Netstumbler, Airsnort, Ethereal) and the car of your choice

- Drive around, use Netstumbler to map out active wireless networks and (using GPS) their access points
- If network is encrypted, park the car, start Airsnort, leave it be for a few hours
 - Airsnort will passively listen to encrypted network traffic and, after 5-10 million packets, extract the encryption key
- Once the encryption key is compromised, connect to the network as if there is no encryption at all
- Alternative: use Ethereal (or packet sniffer of your choice) to listen to decrypted traffic and analyze
- Many networks are even less secure

Netstumbler

What is NetStumbler?

NetStumbler is a tool for Windows that allows you to detect Wireless Local Area Networks (WLANS) using 802.11b, 802.11a and 802.11g. It has many uses:

- Verify that your network is set up the way you intended.
- Find locations with poor coverage in your WLAN.
- Detect other networks that may be causing interference on your network.
- Detect unauthorized "rogue" access points in your workplace.
- Help aim directional antennas for long-haul WLAN links.
- Use it recreationally for WarDriving.

Un'antenna gusto barbecue...



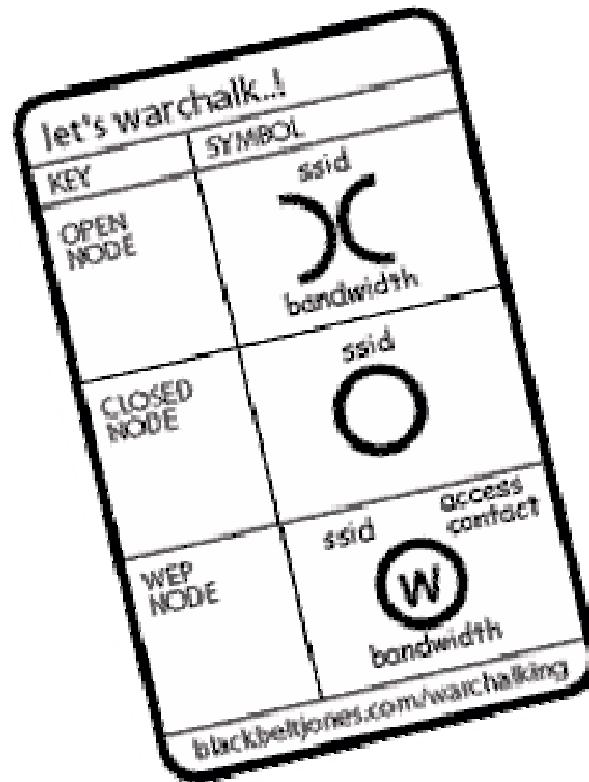
Integrità dei messaggi

- The checksum used by WEP is CRC-32, which is not a cryptographic checksum (MAC)
 - Purpose of checksum is to see if noise modified the message, not to prevent “malicious” and intelligent modifications
- Property of CRC: The checksum is a linear function of the message $c(x \oplus y) = c(x) \oplus c(y)$
- This property allows one to make controlled modifications to a ciphertext without disrupting the checksum:
 - Suppose ciphertext C is: $C = RC4(v, K) \oplus \{M, c(M)\}$
 - We can make a new ciphertext C' that corresponds to an M' of our choosing

Alterazione dei messaggi

$$\begin{aligned} C' &= C \oplus \{\delta, c(\delta)\} \\ &= RC4(v, K) \oplus \{M, c(M)\} \oplus \{\delta, c(\delta)\} \\ &= RC4(v, K) \oplus \{M \oplus \delta, c(M) \oplus c(\delta)\} \\ &= RC4(v, K) \oplus \{M', c(M \oplus \delta)\} \\ &= RC4(v, K) \oplus \{M', c(M')\} \end{aligned}$$

Warchalking...



Contromisure

- Run VPN on top of wireless
 - Treat wireless as you would an insecure wired network
 - VPNs have their own security and performance issues
 - Compromise of one client may compromise entire network
- Hide SSID of your access point
 - Still, raw packets will reveal SSID (it is not encrypted!)
- Have each access point maintain a list of network cards addresses that are allowed to connect to it
 - Infeasible for large networks
 - Attacker can sniff a packet from a legitimate card, then re-code (spoof) his card to use a legitimate address

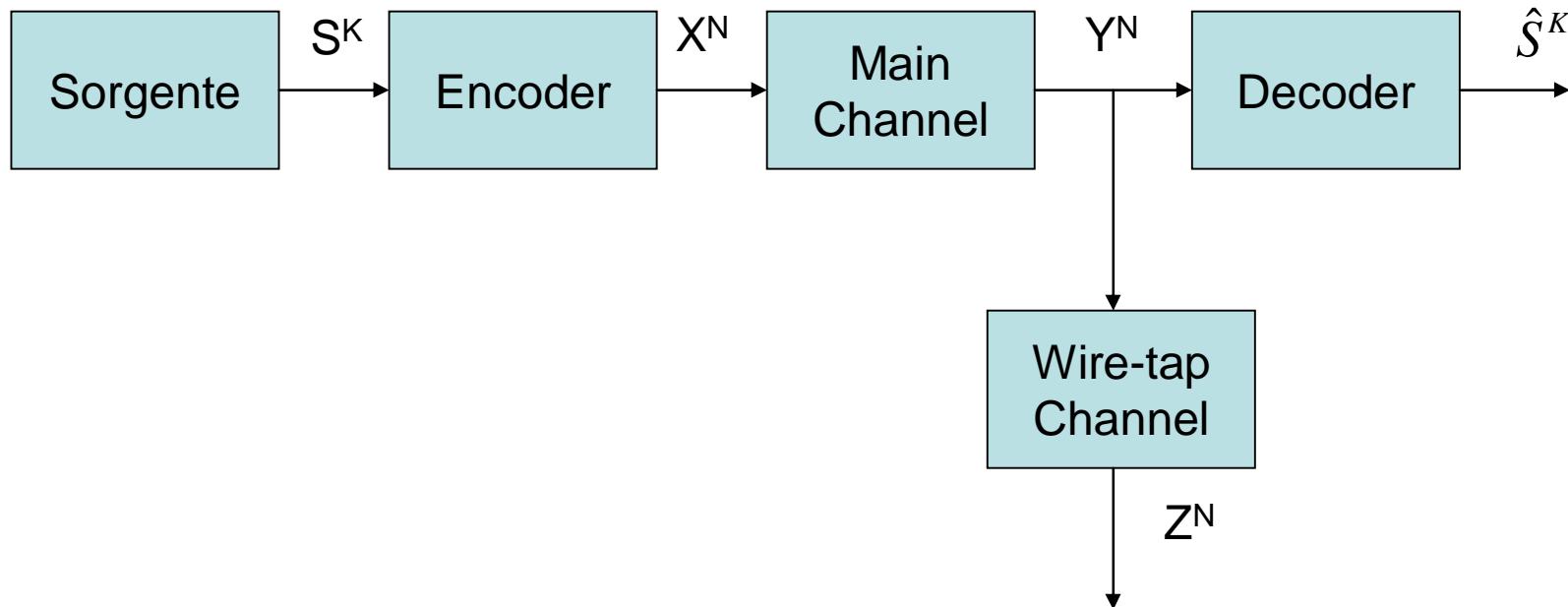
Fixing the Problem

- Extensible Authentication Protocol (EAP)
 - Developers can choose their own authentication method
 - Cisco EAP-LEAP (passwords), Microsoft EAP-TLS (public-key certificates), PEAP (passwords OR certificates), etc.
- **802.11i** standard fixes 802.11b problems
 - Still RC4, but encrypts IVs and establishes new shared keys for every 10 KBytes transmitted
 - No keystream re-use, prevents exploitation of RC4 weaknesses
 - **Use same network card, only upgrade firmware**
 - Long-term: AES, 128-bit keys, 48-bit IVs
 - Block cipher (in special mode) instead of stream cipher
 - Requires new network card hardware

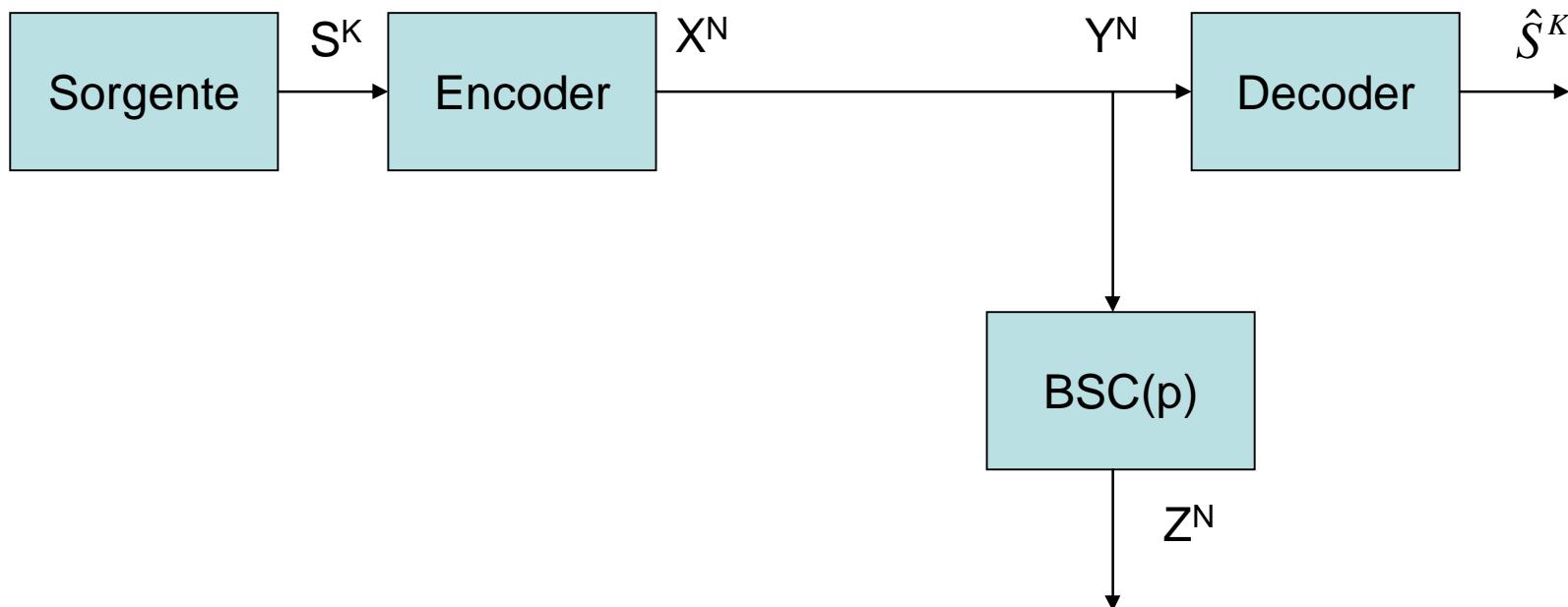
Crittografia al livello fisico

The wire-tap channel

(A.D. Wyner, Oct. 1975)



The BSC wire-tap channel



The Gaussian wire-tap channel

(Leung-Yan-Cheong & Hellman, July '78)

