Crittografia avanzata
Lezione del 7 Marzo 2011

L'informazione nei messaggi (1)

- L'informazione nei messaggi è generalmente ridondante.
- Es. nelle parole di una frase inglese il contenuto di informazione è circa 1.2 bit/carattere
- La ridondanza dell'informazione permette la criptoanalisi

L'informazione nei messaggi (2)

- Gli algoritmi di crittografia mantengono in generale la lunghezza dei messaggi
- Quindi mantengono la ridondanza
- Per non essere crittoanalizzabili, devono rendere difficile “trovare” la ridondanza
  -

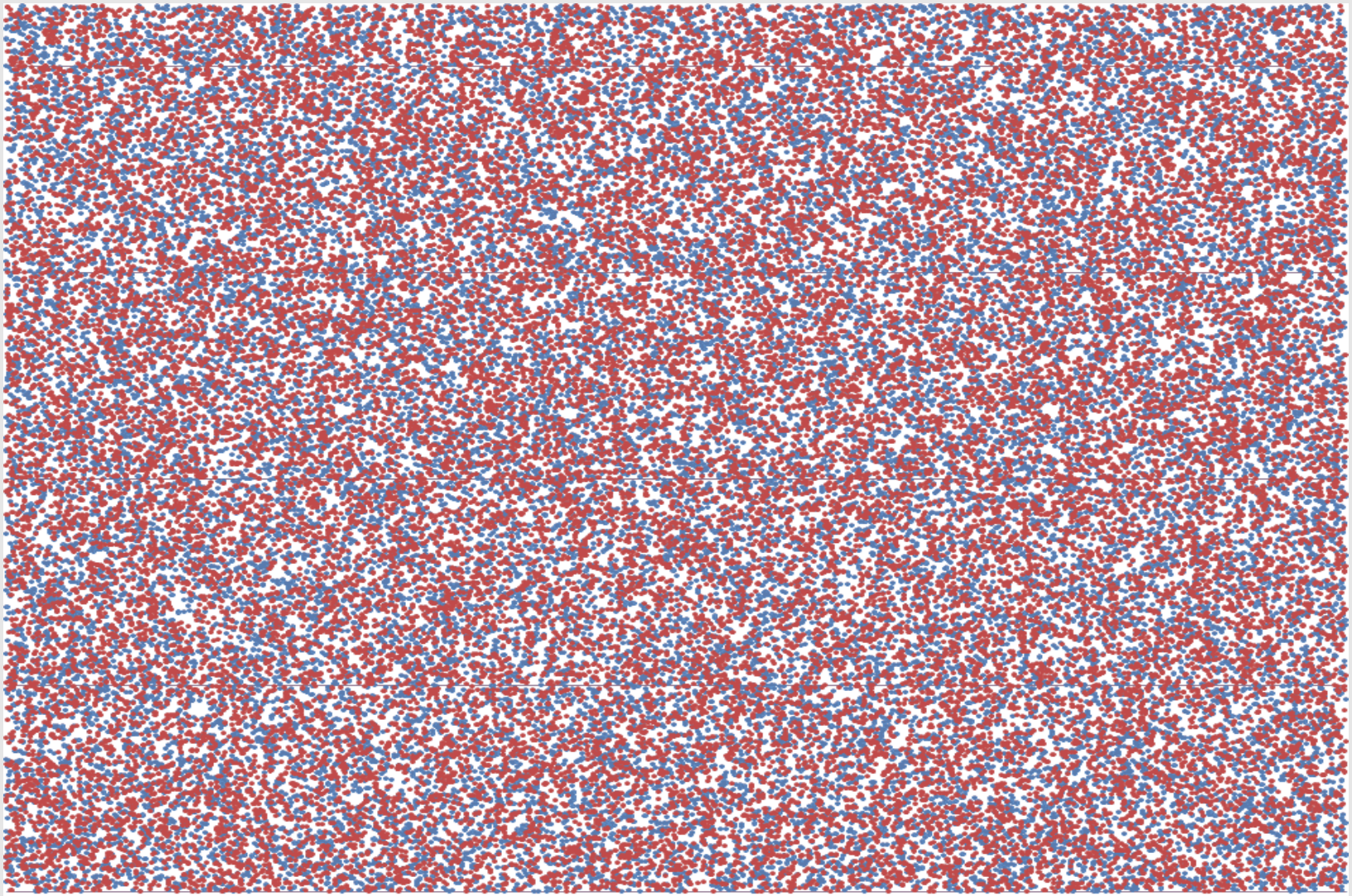
Quattro colori: decifratura

- Supponiamo cifrare con una chiave di 8 bit (256 combinazioni);
- Se proviamo una chiave sbagliata con la codifica a lettere abbiamo :
  - 3/(2^40) probabilità di “collidere” con un colore sbagliato
  - 1-1/(2^38) ~ 1 di avere una stringa senza senso
- Con la codifica a due bit tutte le chiavi danno un valore ragionevole

Qualche numero

- Una pagina di testo ha circa 80 x 24 battute (1920 caratteri) ovvero 15360 bit
- esistono 2^15360 possibili pagine
- secondo il calcolo di 1,2 bit a carattere sono 2304 bit di informazione; ci sono 13056 bit ridondanti
- solo una pagina su 2^13056 è “sensata”
- da una pagina cifrata con una chiave di 128 bit possono essere generate 2^128 diverse pagine “originali”
- qual’è la probabilità di generare una pagina “sensata”che non sia quella originale?
- Come riconoscere che una pagina è sensata?

Casuali o pseudo casuali?



Cattivi generatori

- Generano numeri che vorrebbero essere casuali, ma:
  - Non hanno a disposizione abbastanza entropia, e quindi sono (in parte) prevedibili, ad esempio basati su orologio, temporizzazioni interne...
    - Dov'è l'entropia in un computer?
  - Non producono una distribuzione uniforme, e quindi alcuni numeri sono più probabili

Esempio: RC4

- Sviluppato nel 1987. Nel 1994 l'algoritmo è stato pubblicato anonimamente su Cypherpunks
  - Non si sa da chi, si è sospettato fosse un "dipendente scontento" di RSA
  - Fortunatamente, l'algoritmo era abbastanza valido, perché era usato molto diffusamente...
  - Sono state trovate delle vulnerabilità se non è implementato "correttamente"...
    - Come nelle specifiche di WEP, ad esempio

Quanto deve essere lunga la chiave?

- La potenza computazionale dei computer è in continuo aumento
- Quanto tempo deve rimanere segreto il messaggio?
- Quante risorse possono essere impiegate nell'attacco?
- Naturalmente se la chiave è casuale
- Bruce Schneier: “Nessun ladro prova tutte le chiavi”

Meet-in-the-middle

- Per allungare la chiave lunga n, perché non cifrare semplicemente più volte, usando diverse chiavi? Attacco di choosen/known plaintext:
- $C=E_{k1}(E_{k2}(M))$, k1 e k2 sono “troppo brevi”
- Trade-off fra spazio di memoria e computazione:
  - Si costruisce una tabella di $E_k(M)$ per un testo scelto e per ogni k
  - Si calcola $D_k(C)$ per ogni K; ogni caso in cui il risultato è in tabella ci può aver dato K1 e K2
- Invece di un costo 2^{2n} abbiamo un costo 2^{n+1}

Trade off fra tempo e spazio

- Se è possibile precomputare una parte dell'algoritmo, la lunghezza della chiave risulta ridotta e quindi si riduce il tempo necessario per un attacco di forza bruta
- Serve spazio per la tabella
- È una tecnica molto utilizzata per ridurre la complessità degli attacchi
- PSPACE = spazio dei problemi risolvibili in tempo anche esponenziale ma con spazio polinomiale (da un computer classico)
- PSPACE include NP

Weak keys

- Alcuni algoritmi hanno delle “chiavi deboli”, chiavi che fanno comportare l'algoritmo in modo “strano” e indesiderato:
- DES: alcune chiavi fanno sì che cifrando il testo cifrato, si ottenga il testo in chiaro
- RC4: con almeno 1/256 chiavi, la correlazione fra chiave e testo cifrato è tale da ridurre il costo di ricerca della chiave
- ...

One Time Pad: l'algoritmo perfetto (1)

- Si deve trasmettere in modo sicuro un messaggio di n caratteri
- Si genera una sequenza casuale di n byte
- Si comunica al destinatario la sequenza
- Si calcola l'XOR delle due sequenze
- Si trasmette il risultato

Finti OTP

- Si usa una chiave casuale “di lunghezza arbitraria” K bit
- Un messaggio lungo nK viene cifrato usando N volte la chiave K in XOR
- $M=M_1\ M_2\ \ldots\ M_n$
- $C=C_1\ C_2\ \ldots\ C_n=M_1 \oplus K\ M_2 \oplus K\ \ldots\ M_n \oplus K$
- $C_1 \oplus C_2 = M_1 \oplus M_2$
- Con una tecnica simile si scopre, prima, la lunghezza di K

Gli altri algoritmi

- Utilizzano una chiave di lunghezza fissa, “breve”
- Non tutti i messaggi sono equiprobabili
- Ci si basa sulla lunghezza della chiave per garantire che non sia computazionalmente possibile scoprire il messaggio
  - Tuttavia, il fatto che la chiave sia riutilizzata continuamente è alla base di molti attacchi, soprattutto di known plaintext

Algoritmi a blocchi e "stream"

- Gli algoritmi a blocchi operano su blocchi di n bit per volta
  - a questa categoria appartengono gli algoritmi più noti
- Gli algoritmi "stream" operano su un bit per volta
  - adatti per crittografare flussi (stream) di dati, es. crittografia in linea

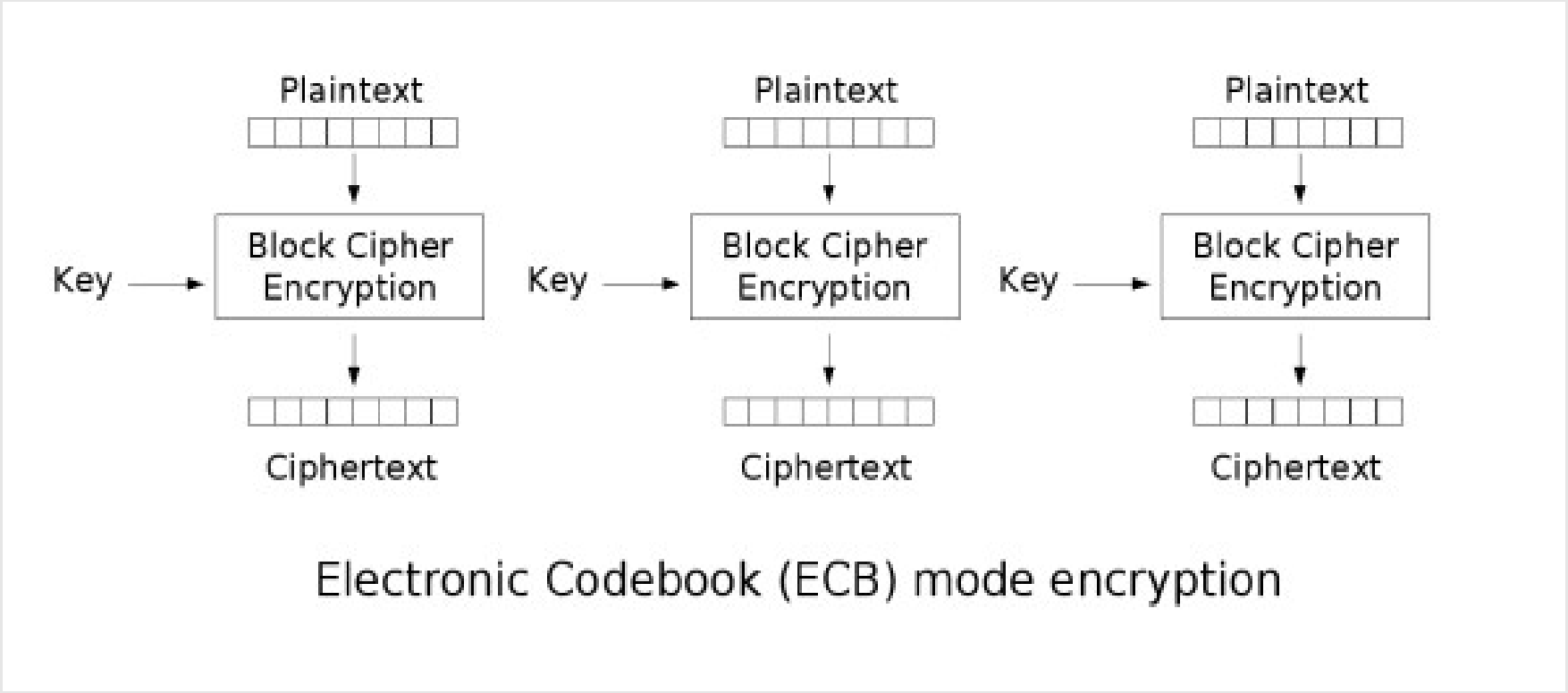
Bit flipping attack

- Si prende uno messaggio cifrato C; si conosce la struttura del testo in chiaro M (messaggio, pacchetto, ecc.)
- Si cambiano bit in punti strategici di C, in modo da modificare corrispondentemente M senza conoscere M
- Funziona se c'è poca diffusione
  - Utilizzato contro WEP

Composizione dei blocchi

- Gli algoritmi a blocchi lavorano su un blocco per volta: come si opera su testi più lunghi?
  - Si dividono in blocchi
  - Si opera sui blocchi...
- Modi di operazione principali
  - Electronic Codebook: ECB
  - Cypher block chaining: CBC
  - Output Feedback: OFB
  - Cypher feedback: CFB

Electronic codebook



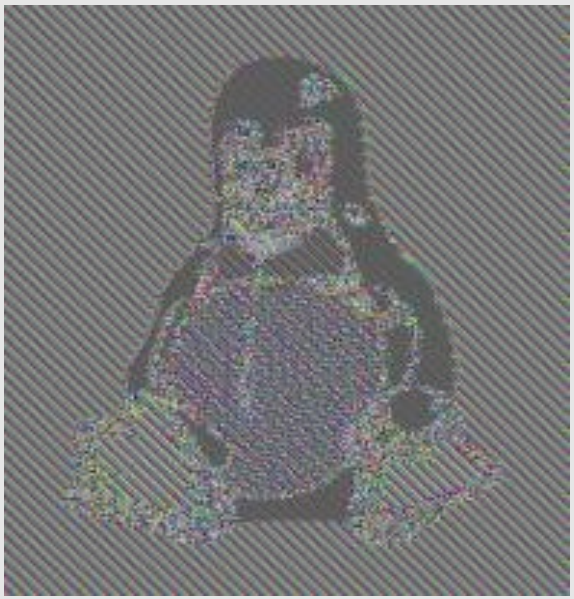
Electronic Codebook (ECB) mode encryption

- I blocchi sono solo concatenati
- Blocchi uguali danno testi cifrati uguali
- Da qui il nome...

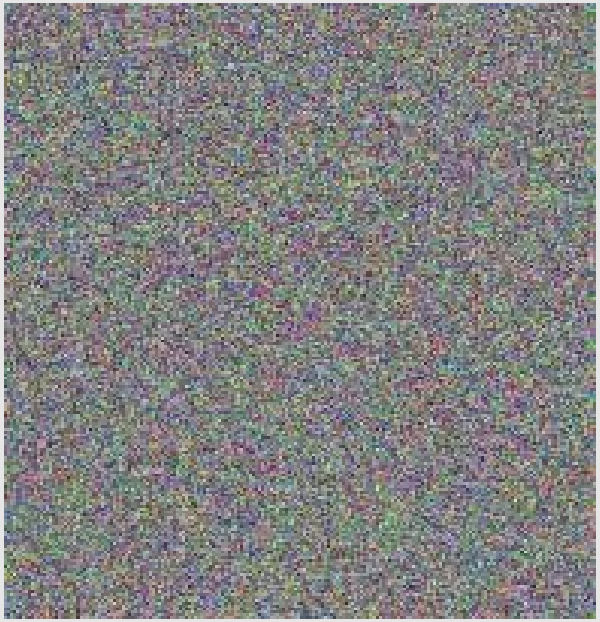
Esempio di ECB

Testo in chiaro

ECB



Altri modi di operazione



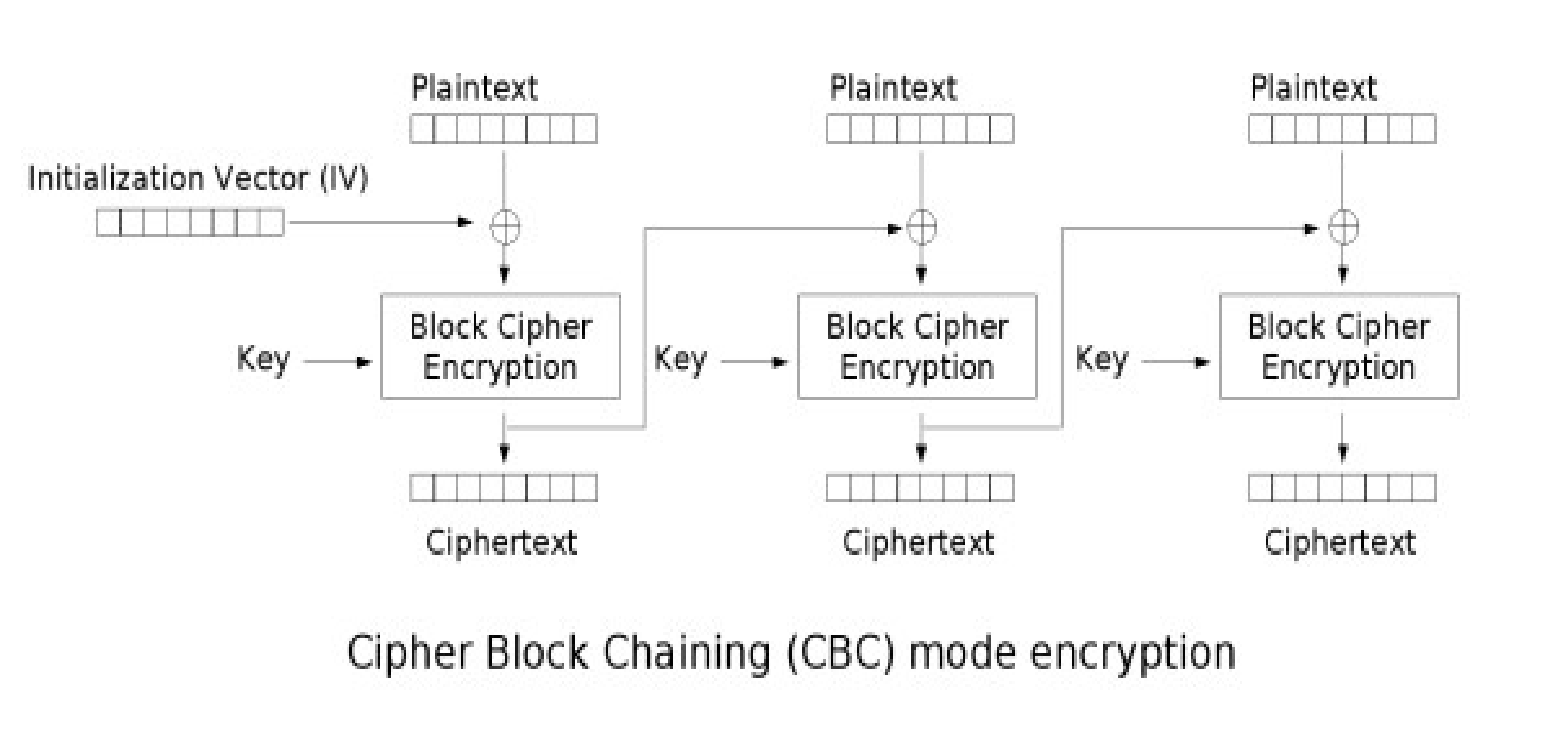
Quando usare ECB?

- Riduce l'impatto degli errori al singolo blocco, ma difficilmente basta per giustificarne l'uso
- Problemi simili: cifratura di singoli campi e altri “messaggi brevi”
  - Non serve l'intero codebook, generalmente basta trovare similitudini
  - Replay attack “ciechi” ripetendo i blocchi

Problema generale dei dati cifrati

- Es. Campi di un record di database
- Nome, Cognome, dati sanitari
- Anonimizzazione: i dati sanitari sono leggibili, nome e cognome sono cifrati
  - Ma ogni campo nome è uguale se il nome è uguale
  - Lo stesso per il cognome
- Lo stesso per gli hash delle password
  - Soluzioni? Es. *salting*

Cypher Block chaining



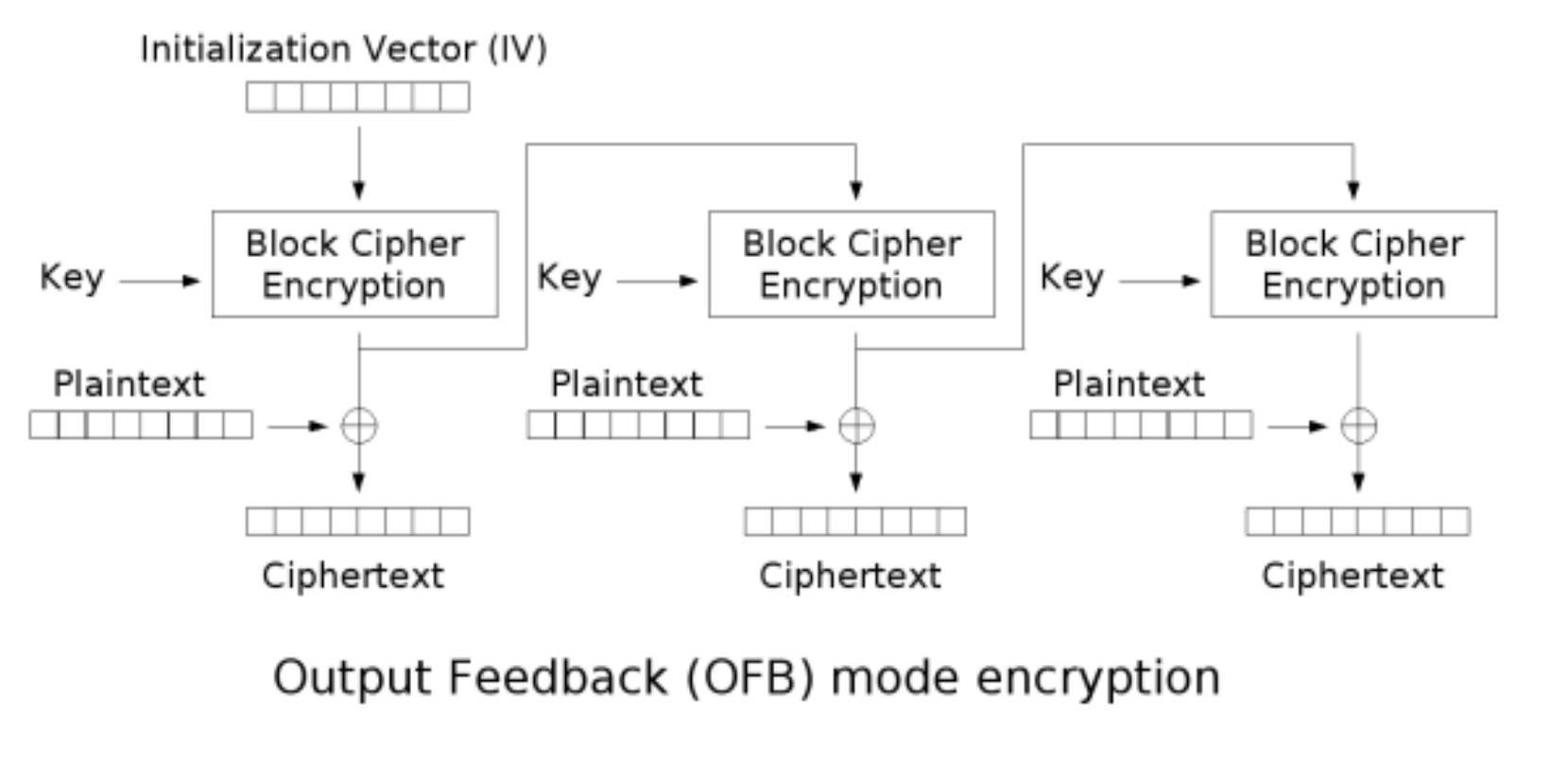
Cipher Block Chaining (CBC) mode encryption

- I blocchi vengono concatenati, quindi il risultato dipende dai blocchi precedenti
- Serve un IV (non segreto)

Altri modi

- Di solito molto simili a CBC
- CFB/OFB: usati per trasformare algoritmi a blocchi in algoritmi stream
- Alcuni sono specializzati
  - per la cifratura dei dischi
- Di solito si usa CBC; se ci sono motivi per non usarlo, si valutano caso per caso le altre opzioni

Output feedback

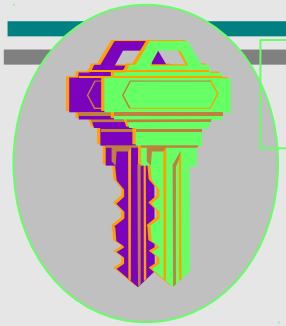


Rende l'algoritmo un algoritmo stream: i bit del plaintext possono uscire come ciphertext a mano a mano che arrivano

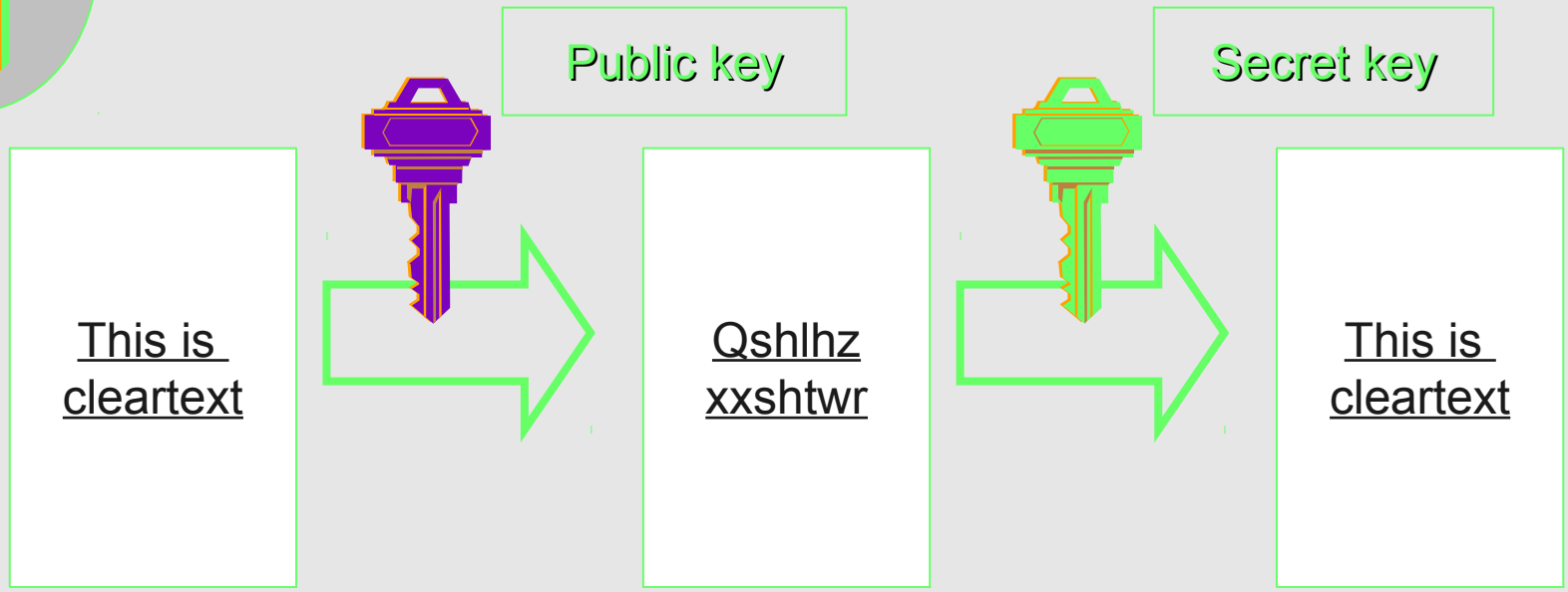
Problemi della crittografia simmetrica

- Le comunicazioni riservate fra n entità richiedono n!/2 diverse chiavi
  - troppe se n è alto (e c'è un punto di centralizzazione)
- Come gestire le comunicazioni con entità "nuove" senza un accordo diretto preventivo?
- Mittente e destinatario conoscono la chiave, quindi non è possibile la *non repudiation*

Algoritmi asimmetrici



Le chiavi sono generate in coppia



- Si utilizzano due chiavi diverse, una per cifrare ed una per decifrare: $M=D_{k1}(E_{k2}(M))$
- k1 e k2 sono generate in coppia, conoscendo l'una non si riesce a risalire all'altra (problemi difficili)

Vantaggi e svantaggi

- Una chiave è pubblicata (la chiave pubblica) e una è segreta
- per le comunicazioni riservate fra n persone servono 2n chiavi
- la chiave pubblica può essere comunicata da terzi (e certificata)
- è possibile la non repudiation
- è molto più lenta della crittografia simmetrica (ora)

Esempio: RSA

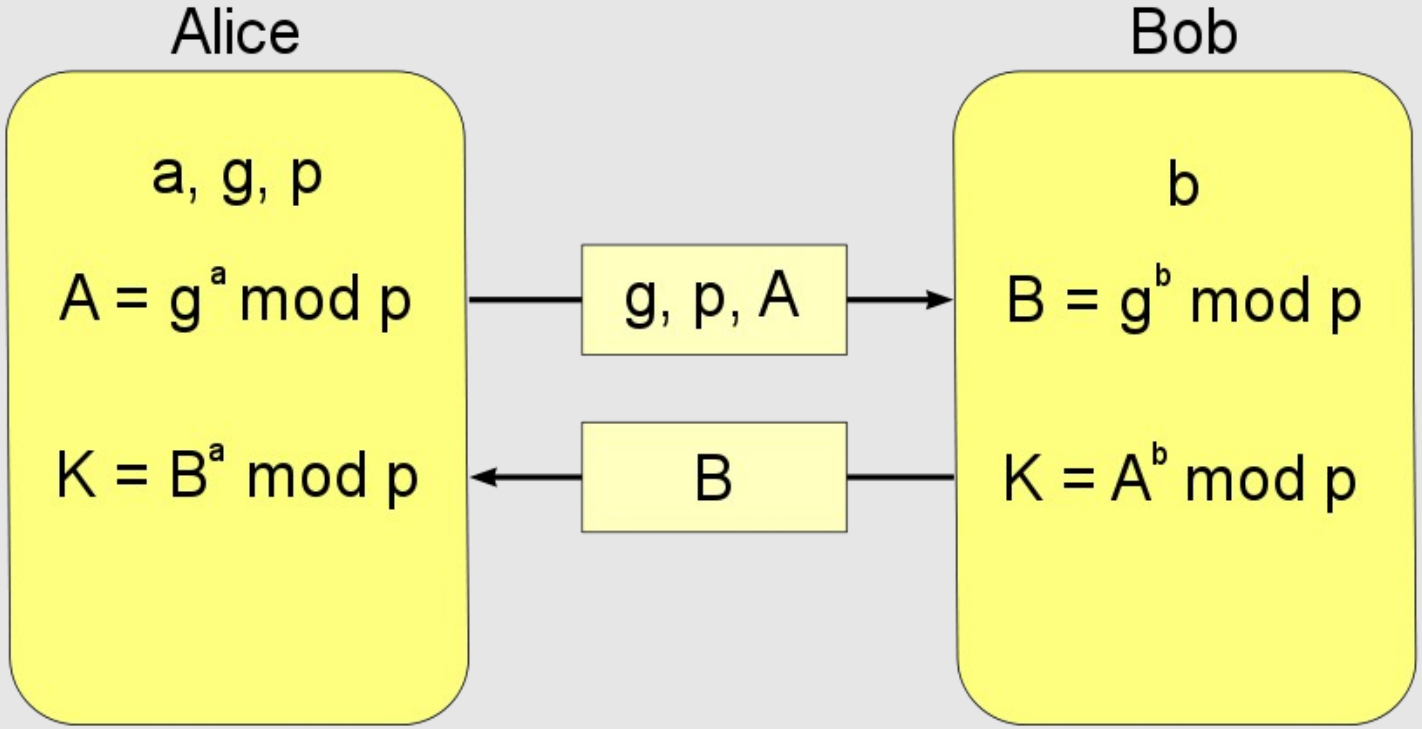
- Utilizza abitualmente chiavi di 512, 768 o 1024 bit
- E' circa due ordini di grandezza più lento di DES
- È ancora uno dei più utilizzati
- Basato sulla difficoltà nella fattorizzazione in numeri primi
  - La generazione di una coppia di chiavi richiede due numeri primi “grandi”, vengono valutati con verifiche statistiche

Timing attacks

- Possibili in tutti i casi in cui le operazioni “segrete” non richiedono un tempo costante (es. decifratura)
  - Misurando il tempo necessario, è possibile avere informazioni sul segreto (es. la chiave)
  - Es. numero di bit a 1 nella chiave
- Nel 1995 è stato descritto un timing attack che permette di recuperare la chiave di decifratura
- Nel 2003 è stato usato con successo contro SSL con RSA
  - Sfruttando la differenza di tempo fra diverse tipologie di errore nella connessione

Diffie-Hellman key exchange

- Come scambiare una chiave su un canale osservato
- Ci si accorda su un primo p e una base g. Poiché $g^{ab} \bmod p = g^{ba} \bmod p$, arrivano alla stessa chiave K



$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

Problema del MitM

- Dato che le due parti non hanno un accordo di alcun tipo:
  - A pensa di condividere una chiave K1 con B, in realtà la condivide con E
  - B pensa di condividere una chiave K2 con A, in realtà la condivide con E
  - A manda il dato cifrato con K1, E lo decifra, lo legge e lo ricifra con K2

MitM

- Il problema è sempre presente quando non ci sono altri accordi tramite canali sicuri
- In teoria, quindi, sempre, perché come si crea il canale sicuro?
  - Problema di identificazione: in realtà ci si assicura che l'entità sia la stessa con cui si sono avuti contatti precedentemente
- Tutto questo ci riporterà alle Certification Authorities

Fattorizzazione di RSA

- Quanto deve essere lunga la chiave?
- Finora è stata fattorizzata una chiave di 663 bit, in un tempo pari a 75 anni di un singolo Opteron 2.2GHz
- Dato l'uso di queste chiavi, si tende a stare “molto abbondanti”, 1024 o 2048 bit

Affidabilità di algoritmi asimmetrici

- La crittografia asimmetrica si basa su problemi “difficili”
- Possono smettere di essere difficili?
- Problema di complessità: problemi in P e problemi in NP
- Quantum computing

P=NP

- Anche nella recente dimostrazione al riguardo (P≠NP) proposta nell'estate 2010 sono stati trovati degli errori (Febbraio 2011) e per ora non ha dimostrato niente
- Quanto è importante da un punto di vista pratico?
  - Un problema O (n^3) è già di fatto difficile da trattare nella pratica in molti casi (es. ordinamenti)
  - Non si possono trascurare i fattori costanti (di grande impatto ad es. nelle fattorizzazioni riuscite di RSA)

Quantum computing

- Esistono dei problemi per i quali l'utilizzo di un computer quantistico sembra(va) offre dei vantaggi
- Alcuni algoritmi asimmetrici rientrano fra questi
  - es. basati sulla difficoltà nel fattorizzare grandi numeri
  - E quindi la realizzazione di computer quantistici di uso pratico ne mettesse a rischio l'efficacia
  - È stato recentemente dimostrato che in realtà l'uso di computer quantistici non offre vantaggi (in termini di complessità) per problemi in IP, e quindi anche in NP

Funzioni Hash sicure

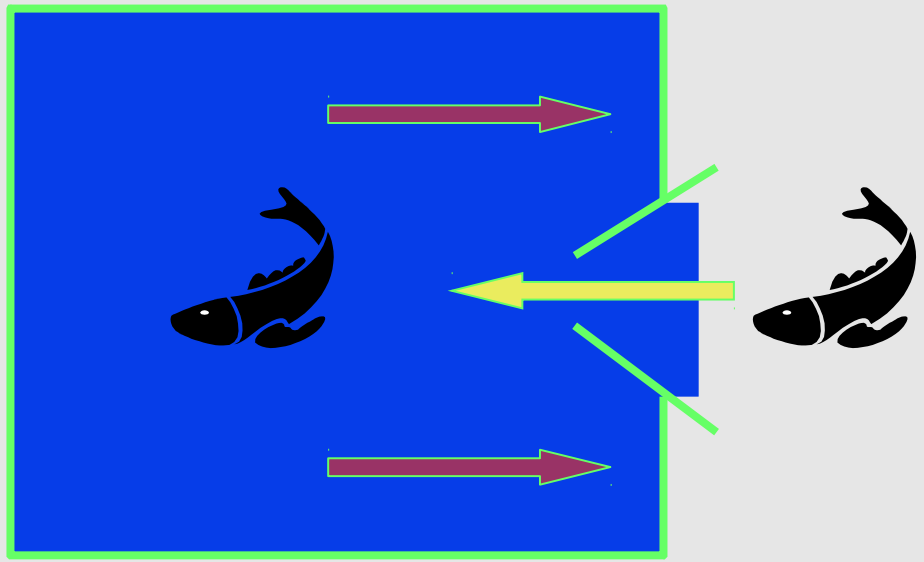
- Dato un testo di lunghezza arbitraria restituiscono un digest breve e di lunghezza fissa
- Dato un digest, non è (computazionalmente) possibile risalire al testo originale
  - E/ma in generale non sono biunivoche
- Dato un digest D, non è possibile costruire un testo M che generi come digest D
- Questo vuole dire che è difficile modificare una stringa senza modificare il digest (controllo di integrità)

One-Way Hash functions

This is cleartext

iaxjfhzz

Basket-net



Cleartext to digest is easy

Digest to cleartext is hard

Esempi: MD5 e SHA

- Entrambe derivano da MD4
- MD5:
  - Produce un hash di 128 bit
- SHA-1:
  - Produce un hash di 160 bit
- Sono entrambe funzioni ormai deboli, vengono man mano abbandonate ma finora sono probabilmente ancora le più usate

Esempi: SHA-2 e RIPEMD-128

- SHA-2: al momento è la funzione “di scelta”
  - Chiave da 224 a 512 bit
- RIPEMD-128
  - Chiave da 128 a 320 bit (RIPEMD-160...)
  - Famiglia diversa dall'originale RIPEMD, per il quale si conoscono collisioni

Collisioni

- È la tipologia principale di attacchi alle funzioni hash
  - Trovare due testi che abbiano lo stesso hash
    - Interessante dal punto di vista teorico (mostra la debolezza dell'algoritmo, di solito poco interessante dal punto di vista pratico)
  - Dato un testo e un hash, trovare un altro testo generico che abbia lo stesso hash e sia “valido”
  - Dato un testo e un hash, trovarne un altro simile che abbia il “significato” desiderato e lo stesso hash

Birthday attack

- Paradosso dei compleanni: Qual'è la probabilità che in un gruppo di n persone, due abbiano il compleanno lo stesso giorno?
- $365!/(365^n(365-n)!)$ es. 50% su 23 persone, 100% su 366 persone
- Dati due testi T1 e T2, uno vero e uno falso, si generano T1' e T2' che variano rispetto agli originali per spazi ecc. (non rilevanti)
- si calcolano gli hash e si confrontano quelli di ogni variazione di T1 con quelli di ogni variazione di T2
- Con al più $2^{m/2}$ coppie si trova una collisione

Esempi: MD5

- Nel 2005 sono stati create coppie di documenti PS con lo stesso hash
  - Il trucco sta nel nascondere le modifiche necessarie in stringhe non interpretate dal viewer ps
- Nel dicembre 2008, sono stati modificati certificati in modo da aggiungere la proprietà di essere di Certification Authority
  - sfruttando debolezze di MD5
  - Usando un cluster di Playstation

Concorso per una nuova funzione hash

- Per qualche motivo, nonostante i “segni” di problemi, c'è stato a lungo poco interesse per le funzioni hash
- Al momento, ci sono poche opzioni valide; il NIST ha lanciato un concorso, simile a quello che ha portato ad AES, per una funzione hash; ci si aspetta un risultato per il 2012
- Sono stati selezionati da poco i cinque finalisti

Controllo dell'integrità

- Dato un messaggio M, se ne calcola il digest D mediante una funzione hash H
- D = H(M)
- La trasmissione **fuori linea** del digest D garantisce la correttezza del messaggio
  - Scaricare il digest insieme al messaggio/file/codice non fornisce di per sé un controllo di integrità, dato che possono essere modificati contestualmente

Permutazione

- Dato un testo T, il testo cifrato C è il risultato di una permutazione per blocchi di dieci lettere:
  - es. con chiave 1509346728 il testo “abcd fghil” diventa “a licdfgbh”
  - Ci sono 10! possibili permutazioni, ovvero circa 22 bit di chiave
- Il testo cifrato è

R	D	S	C	O	I	O	C	O	
T	R	T	I	I	A	F	R	O	G
A	Z	V	A		A	T	A	A	N

Soluzione

1	2	3	4	5	6	7	8	9	0
3	7	4	0	2	8	9	1	5	6
C	O	R	S	O		D	I		C
R	I	T	T	O	G	R	A	F	I
A		A	V	A	N	Z	A	T	A
R	D	S	C	O	I	O	C	O	
T	R	T	I	I	A	F	R	O	G
A	Z	V	A		A	T	A	A	N

Considerazioni

- Nonostante la chiave di 22 bit, il testo in chiaro “traspare” tanto da permettere analisi molto più semplici della forza bruta
  - es. la frequenza delle lettere non cambia
  - Su un blocco è possibile cercare (frammenti di) parole e lavorare in modo combinatorio
  - Il fatto che sia ECB aiuta la criptoanalisi anche se non ci sono due blocchi uguali

Algoritmo 2

- Non sapete l'algoritmo
- Non sapete la chiave
- Ma la chiave è di due cifre decimali, per un totale di 100 possibili chiavi
- Il testo è

  PKN SGBFQ FKN ICSOOP JK TQYZXG ZOVG

Soluzione

- Algoritmo tipo cifrario di Cesare (sostituzione) con trasposizione alternata di 2 o 7 lettere (chiave 2,7)

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

Considerazioni

- Nonostante l'algoritmo banale e la chiave corta, fra conoscere l'algoritmo e non conoscerlo c'è differenza...
- Qui è come avere due “mezzi messaggi” cifrati separatamente, e anche qui le frequenze delle lettere traspaiono (compare due volte la mappatura e->k in parole brevi...)