

Crittografia e sicurezza delle reti

Crittografia simmetrica

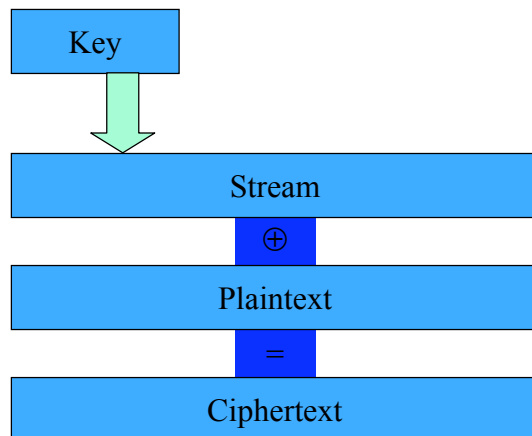
- Stream & Block Ciphers
- Modalità di codifica
- Gruppi, Anelli e Campi di Galois
- AES

Stream Ciphers

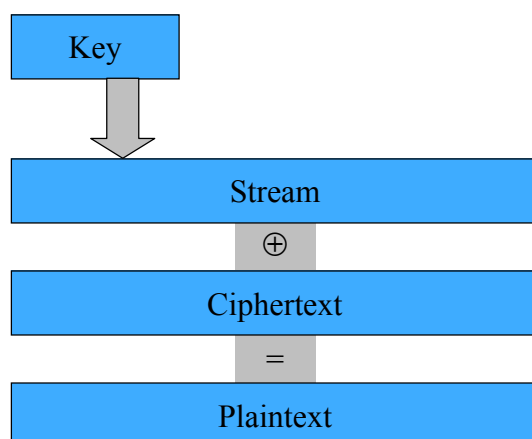
- Inizia con una chiave segreta ("*seed*")
- Genera uno stream di byte (*Keystream*):
 - byte i dello stream è funzione della chiave (Stream cypher *sincrono*) oppure
 - della chiave e dei primi $i-1$ byte del testo crittato (Stream cypher *asincrono*).
- Combina lo stream con il testo in chiaro per ottenere il testo crittato (tipicamente con *XOR*)

si cerca di simulare il codice one-time-pad

Stream Cipher Binari e additivi



Decodifica



Cipher Streams in pratica

- Molti codici prima 1940
- Enigma
- A5 - codifica GSM verso stazione base
- WEP - codifica per 802.11
- RC-4 (Ron's Code)

Esempio: RC-4

- Parte della famiglia RC (Ron's Code, Ron=Ronald Rivest)
- Considerato sicuro: fra 1987 e 1994 il suo comportamento interno non rilevato, dopo molto studiato
- Favorito nelle esportazioni
- Codice molto semplice disponibile
- Molto usato: Lotus Notes, SSL, Wep etc.

RC4: proprietà

- Dimensione chiave variabile (orientata ai byte)
- Sincrono
- Genera una permutazione che appare casuale; la permutazione è ciclica con un periodo molto lungo (più di 10^{100}) [*in questo modo si cerca di simulare il codice one-time-pad*]
- Efficiente: 8-16 operazioni per byte in output

RC-4 Inizializzazione

1. $j=0$
2. $S_0=0, S_1=1, \dots, S_{255}=255$
3. Sia la chiave (bytes) k_0, \dots, k_{255} (ripetere bit se necessario)
4. For $i=0$ to 255
 $j = (j + S_i + k_i) \bmod 256$
Scambia S_i e S_j

In questo modo si ottiene una permutazione di $S_0=0, S_1=1, \dots, S_{255}=255$, funzione della chiave

RC-4 Creazione Key-stream

Genera un output byte B del Key-stream:

1. $i = (i+1) \bmod 256$
2. $j = (j + S_i) \bmod 256$
3. Scambia S_i e S_j
4. $t = (S_i + S_j) \bmod 256$
5. $B = S_t$

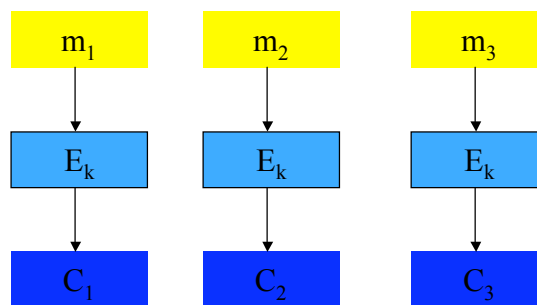
Codifica di un byte del messaggio: B è in XOR con il prossimo byte del testo in chiaro

Modalità di codifica

Codifica Blocchi

- Codifica un blocco di input in un blocco di output
- Tipicamente due blocchi hanno la stessa lunghezza (DES= 64, AES= 128)
- Differenti modi per codificare un testo più lungo di un blocco
- DES, 3-DES, AES (Rijndael), RC-2, RC-5, IDEA, Blowfish, Cast, Gost

Codifica ECB (Electronic Code Book)



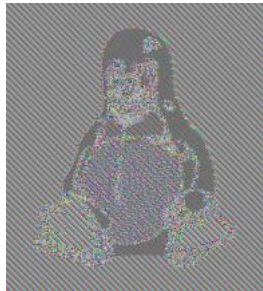
codifica ogni blocco separatamente

Codifica ECB: ripetizioni

originale



codifica ECB



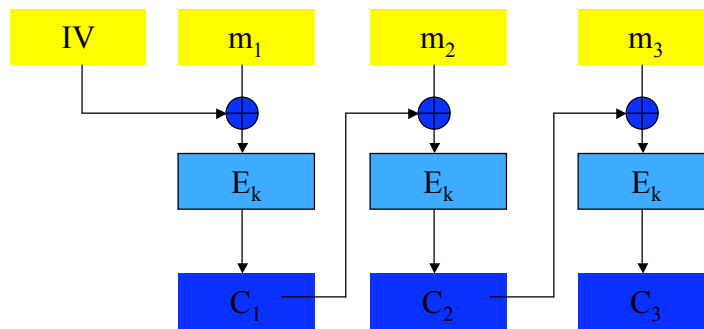
codifica sicura



Proprietà di ECB

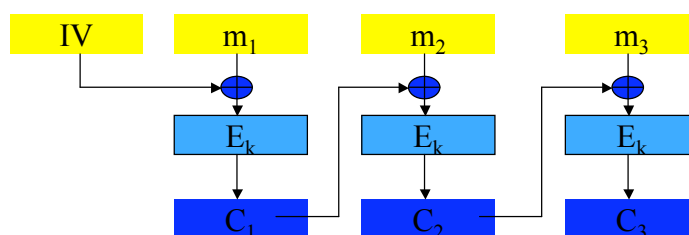
- Semplice e efficiente
- Implementazioni parallele
- ★ Non cancella ripetizioni del testo in chiaro
- Possibili attacchi: (rimuovere, ripetere o scambiare blocchi di codice)

Codifica CBC (Cipher Block Chaining)



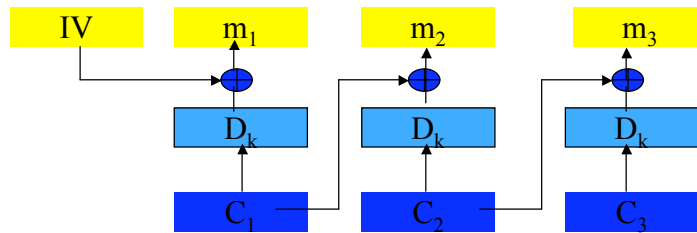
Si utilizza un vettore di inizializzazione IV come seme iniziale del processo "seed"

Codifica CBC (Cipher Block Chaining)



Seed può essere trasmessa in chiaro
Seed = 0 va bene in molti casi; (ma se un file viene trasmesso una volta a settimana, es. stipendi, allora ci accorgiamo di modifiche di file), quindi meglio random

Decodifica CBC (Cipher Block Chaining)



Problema:

cambiando bit in $C(i-1)$ si modifica $m-i$ in modo prevedibile,
ma questo provoca cambiamenti in $m-(i-1)$ non prevedibili;

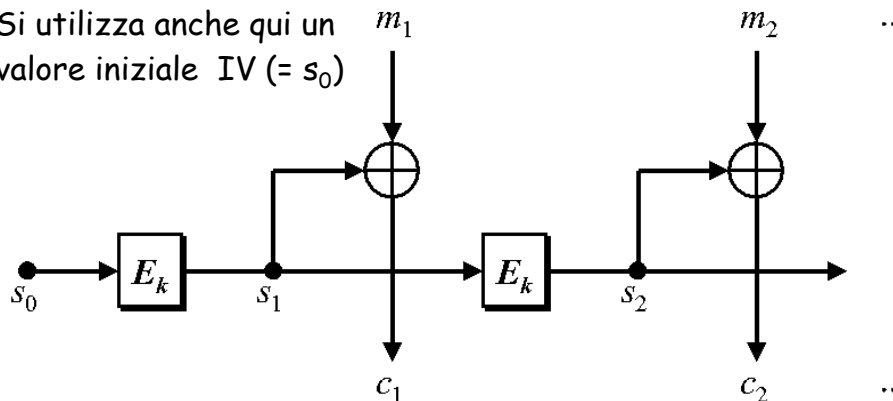
soluzione: aggiungere controllo errori trasmissione (CRC)

Proprietà di CBC

- Stream cipher asincrono
- Errori si propagano nel blocco successivo
- Problemi con perdite di bit del testo codificato
- Cancella ripetizioni del testo
- Non sono note implementazioni parallele
- Il testo in chiaro non può essere facilmente modificato
- Standard in molti sistemi: SSL, IPsec etc.

Codifica OFB (Output FeedBack)

Si utilizza anche qui un valore iniziale IV ($= s_0$)



Obiettivo: generare con f una sequenza che appaia casuale come la stringa usata nel codice one-time-pad

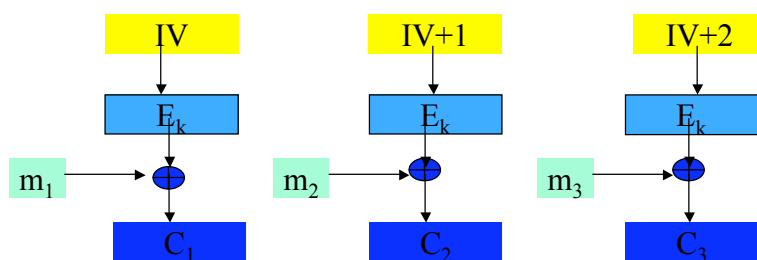
Codifica OFB (Output FeedBack)

- se la funzione f è pubblica (nota ad un attaccante) allora il valore iniziale s_0 trasmesso all'inizio deve essere protetto (perchè?)
- se la funzione f è crittografica e dipende da una chiave segreta allora il valore iniziale può essere trasmesso in chiaro (perchè?)
- il valore iniziale deve essere modificato per ogni messaggio anche se f è crittografica e dipende da chiave segreta (se si conosce una coppia messaggio, valore iniziale un attaccante è in grado di codificare un qualunque messaggio. perchè?)
- si può generalizzare al caso in cui solo alcuni bit dello stream sono utilizzati per il calcolo del codice cifrato (k-OFB)

Proprietà di OFB

- Stream cipher sincrono
- Errori nel testo codificato non si propagano
- Problemi con perdite di bit del testo codificato
- Possibile utilizzare preprocessing
- Cancella ripetizioni del testo
- Non sono note implementazioni parallele
- Sono possibili attacchi manipolando il testo in chiaro

CTR (Counter Mode)



simile a OFB calcolato prima
come OFB ha problemi nell'uso ripetuto di IV

vantaggio

puoi calcolare il messaggio ad ogni punto e non dall'inizio

Proposte per AES

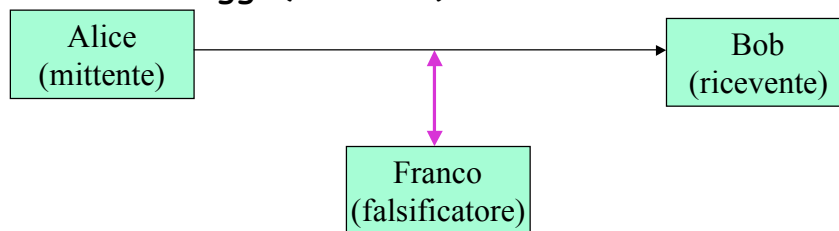
- Modalità CTR (Counter) (modifica di OFB): Implementazione parallela, pre-processing offline, dimostrabile sicuro, semplice e efficiente
- Modalità OCB (Offset Codebook) Implementazione parallela, pre-processing offline, dimostrabile sicuro (sotto specifiche ipotesi), garantisce autenticità del messaggio

Crittografia e sicurezza delle reti

Integrità dei dati e loro autenticità
Message Authentication Codes (MAC)

Scopo

Garantire l'integrità dei messaggi anche in presenza di un avversario attivo che manda i suoi messaggi (corretti)



Nota: **Autenticazione** è ortogonale alla **segretezza**
spesso un sistema deve garantire ambedue

Definizioni

- Algoritmo di autenticazione - A
- Algoritmo di autenticazione - V ("accetta"/"rifiuta")
- Chiave di autenticazione- k
- Spazio dei messaggi (stringhe binarie): ogni messaggio fra Alice e Bob è una coppia
- $(m, A_k(m))$
- $A_k(m)$ è impronta (authentication tag) di m

Definizioni (cont.)

- Requisito: esiste una funzione $V_k V_k(m, A_k(m))$
= accetta/rifiuta
 - La verifica si effettua calcolando l'autentica di m
e confrontando con $MAC_k(m)$
 - L'algoritmo di autenticazione è anche detto **MAC**
(Message Authentication Code) $MAC_k(m)$ denota
 $A_k(m)$

Proprietà funzioni MAC

- Requisito di sicurezza
- **avversario** non può costruire una nuova coppia corretta $(m, MAC_k(m))$ anche dopo aver visto $(m_i, MAC_k(m_i))$ ($i=1,2,\dots,n$)
- Output deve essere corto
- Funzioni MAC non sono 1-to-1

Modello Avversario

- Dati disponibili:
 - algoritmo MAC
 - Known plaintext
 - Chosen plaintext
- Nota: chosen plaintext MAC è realistico?
- Scopo: Date n coppie legali
- $(m_1, MAC_k(m_1)), \dots, (m_n, MAC_k(m_n))$
- trova una nuova coppia legale $(m, MAC_k(m))$

Modello Avversario

L'avversario ha successo anche se il messaggio che trova non ha senso.

La ragione per questo è dovuta al fatto che spesso - in un contesto nuovo

- non è facile distinguere cosa ha significato e cosa no
- il ricevente reagisce a messaggi considerati autentici

Efficienza

Scopo avversario: Date n coppie legali

$(m_1, MAC_k(m_1)), \dots, (m_n, MAC_k(m_n))$

- trovare una nuova coppia legale $(m, MAC_k(m))$
- **efficientemente e con probabilità non trascurabile**
(provare a caso quando il MAC è formato da 160 bit allora si ha una probabilità $1/2^{160}$ di indovinare; troppo bassa)

NOTA: Se n è grande abbastanza allora n coppie

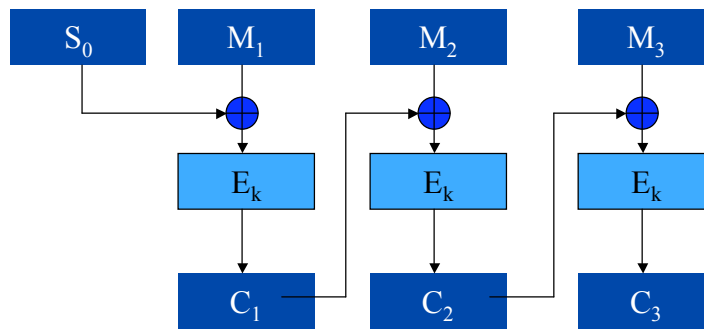
- $(m_i, MAC_k(m_i))$ determinano la chiave k **unicamente** (con alta prob.).
- Quindi un elaboratore nondeterministico può indovinare k e verificare. Eseguire questo deve essere un compito computazionalmente difficile.

MAC Usati in pratica

Vediamo

- **MAC** basato su modalità codifica **CBC**
 - usa codifica crittog. a blocchi
 - lento, problemi esportazione fuori USA
- **MAC** basato su funzioni hash crittografiche
 - veloce
 - nessun problema esportazione

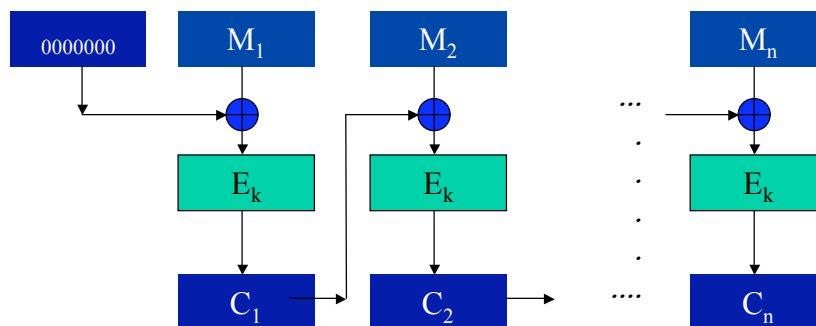
Ricorda: CBC Mode Encryption (Cipher Block Chaining)



Il blocco codificato precedente è in XOR con il corrente blocco in chiaro prima della codifica del blocco corrente
Si utilizza un vettore S_0 di inizializzazione (seed - seme) del processo. S_0 può essere trasmesso in chiaro

MACs con CBC

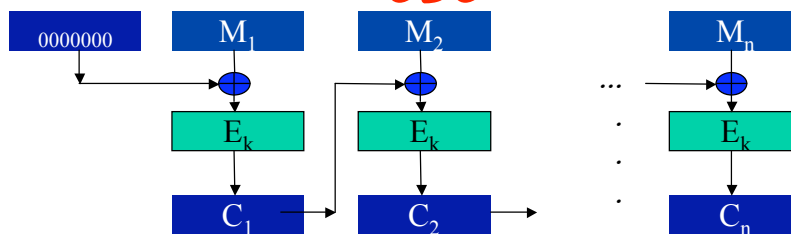
- Inizia con il seme di tutti zero.
- dato un messaggio di n blocchi M_1, M_2, \dots, M_n , applica CBC (usando la chiave segreta k).



- Scarta i primi $n-1$ "ciphertext" blocchi C_1, C_2, \dots , usa C_n .
- Invia M_1, M_2, \dots, M_n & il tag di autenticazione $MAC_k(M) = C_n$.

MACs e confidenzialità con

CBC



è possibile usare CBC per MAC e per segretezza?..

- stessa chiave per inviare messaggio e MAC?

NO (Esercizio: fornire esempi)

- due chiavi distinte: una per confidenzialità del messaggio la seconda per integrità (codifica MAC). Questa soluzione ha un costo maggiore in termini di requisiti (due chiavi) e di calcolo (due calcoli)
 - però le due chiavi possono essere correlate fra loro (originare da un'unica chiave, cambiando alcuni bit della prima per ottenere la seconda, ad esempio in kerberos)

MACs e confidenzialità con CBC

- la ricerca di un metodo per codifica e integrità allo stesso tempo è un problema non completamente risolto (proposte esistenti o non sono sicure o non sono state analizzate a sufficienza)
- una possibile soluzione più veloce della codifica ripetuta con CBC è data dall'uso di una funzione hash crittografica del testo per calcolare un'impronta del messaggio e quindi la sua codifica con ECB
- la soluzione di usare un codice noto come CRC non è considerata sicura (Esercizio: discutere la non adeguatezza di CRC a 32 bit)

Sicurezza di CBC MAC

Proposizione: Se E_k è una funzione **pseudo casuale**, allora CBC MAC è robusto per messaggi di lunghezza fissata

- Cosa significa: se la sequenza di byte generata dalla funzione crittografica è pseudocasuale allora il MAC calcolato con CBC appare come una sequenza casuale; quindi un avversario non può predire il MAC e può solo provare a indovinare il MAC (se fosse in grado di fare qualcosa di più astuto allora sarebbe in grado di predire come è fatta la sequenza pseudocasuale - contraddizione)

Sicurezza di CBC MAC

CBC non è sicuro per messaggi di lunghezza variabile.

Attaccante non conosce la chiave k ma può chiedere di calcolare il CBC MAC di messaggi ed è in grado di calcolare il CBC MAC di un nuovo messaggio.

Attacco: Usa due coppie di messaggio MAC (m, t) e (m', t') di un blocco, e produce un nuovo messaggio $m || (t \text{ EXOR } m')$ (dove $||$ denota concatenazione di stringhe) **che ha come CBC MAC t'** .

- Infatti CBC-MAC di $m || (t \text{ EXOR } m')$ è $E(E(m) \text{ EXOR } (t \text{ EXOR } m'))$, dove $E(m) = t$ e $E(m') = t'$, e quindi $t' = E(m') = E(t \text{ EXOR } t \text{ EXOR } m') = E(E(m) \text{ EXOR } t \text{ EXOR } m')$.

Segretezza & MAC: due chiavi

Dato un messaggio di n blocchi M_1, M_2, \dots, M_n ,
applica CBC (usando la chiave segreta k_1) per
produrre $MAC_{k_1}(M)$

- Produci n blocchi di ciphertext C_1, C_2, \dots, C_n
con una *diversa chiave*, k_2
- Invia C_1, C_2, \dots, C_n e il tag di autenticazione
 $MAC_{k_1}(M)$

CMAC

CMAC: proposta recente (standard NIST 2005)

Modifica proposte precedenti simili

Per generare un MAC CMAC di un messaggio con un codice a
blocchi E di b bit e chiave segreta k

1. utilizza k e genera due chiavi k_1 e k_2
2. usa k_1 e k_2 per calcolare MAC (in modo simile a CBC)

CMAC: generazione chiavi

Genera le chiavi

- 1. Calcola $k_0 = E_k(0)$ (codifica di 000...0 con K).
- 2. se $\text{msb}(k_0) = 0$ allora $k_1 = k_0 \ll 1$ altrimenti $k_1 = (k_0 \ll 1) \text{ EXOR } C$, ($a \ll b$ denota shift a sinistra di a di b posizioni e C è costante ($C = \dots 010000111_2$ per codice 128-bit E, es. AES, e $C = 0\dots 011011$ per 64-bit cipher, es. DES)).
- 3. se $\text{msb}(k_1) = 0$ allora $k_2 = k_1 \ll 1$ else $k_2 = (k_1 \ll 1) \text{ EXOR } C$.

CMAC: calcolo MAC

Il calcolo di CMAC di un messaggio m è:

1. Dividi m in blocchi da b bit $m = m_0 m_1 \dots m_n$ message (m_n non necessariamente di b bit).
2. se m_n è un blocco di b bit allora $m_n = k_1 \text{ EXOR } m_n$ altrimenti $m_n = k_2 \text{ EXOR } (m_n \setminus | 10\dots 0)$.
3. Sia $c_0 = 0$.
4. Per $i = 1, \dots, n$, calcola $c_i = E_k(c_{i-1} \text{ EXOR } m_i)$. (come CBC)
5. Output c_n

Aumentare la sicurezza di un metodo di cifratura

- Progettare un metodo con chiavi di lunghezza variabile - AES
- Whitening - idea di DESX
- Ripetizioni di codifiche (iterated cipher) - Triple DES (3-DES), triple IDEA etc.

Scopo Avversario

- Scopo finale: individuare la chiave
- Obiettivi intermedi:
 - Ridurre lo spazio delle chiavi
 - Riconoscere ripetizioni del testo
 - Determinare parti del testo
 - Modificare il testo cifrato per ottenere testo ragionevole (anche senza rompere il sistema - attacco attivo)

Attacchi Generici

- Ricerca esaustiva
 - Tipo Attacco: ciphertext
 - Tempo: $2^{|k|}$ decodifiche per ciphertext
 - Memoria: costante
- Table lookup
 - Tipo attacco: chosen plaintext
 - Tempo: offline $2^{|k|}$ decodifiche, online costante
 - Memoria: $2^{|k|}$ ciphertexts

Altri attacchi a DES

- Attacchi basati su tradeoff memoria tempo
- Attacchi chosen plaintext
- Linear Cryptanalysis
 - Approssimazione lineare per descrivere DES
 - DES si rompe:
 - 8 rounds: 2^{21} known plaintext
 - 16 rounds: 2^{43} o 2^{47} known plaintext
- Differential analysis
 - Rompe DES con 2^{47} chosen plaintext

Codifiche ripetute

- Per aumentare la lunghezza della chiave il testo in chiaro viene codificato ripetutamente
- Quante volte? 2,3, 678?
- In pratica triple cipher
 - $C = E_{k_1}(E_{k_2}(E_{k_1}(P)))$ [modalità EEE] oppure
 - $C = E_{k_1}(D_{k_2}(E_{k_1}(P)))$ [modalità EDE]
 - EDE è più usato in pratica

Doppio DES: Attacco nel mezzo

Codificare due volte con la stessa chiave?
NO!

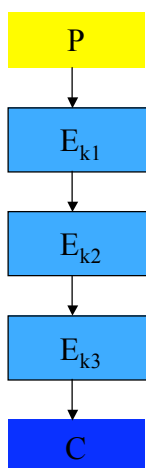
Codificare due volte con chiavi diverse?
No: Attacco nel mezzo richiede

- plaintext noto
- 2^{k+1} codifiche e decodifiche
- $|k|2^{|k|}$ spazio di memoria
- Si utilizza la radice quadrata delle possibili chiavi al costo di memorizzare coppie di <testo, testo codificato>

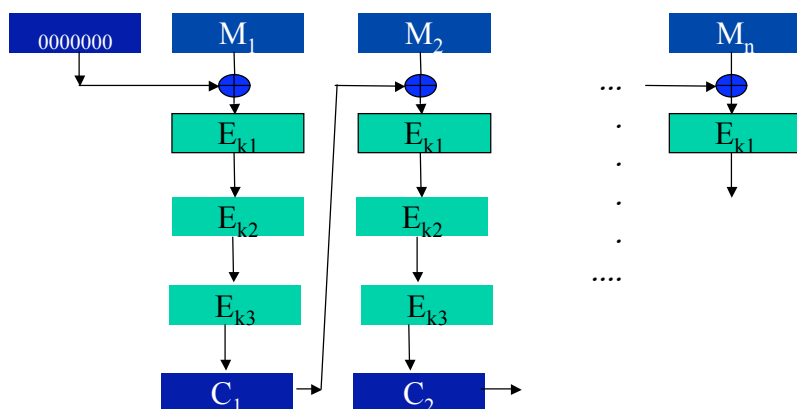
Attacco nel mezzo (cont.)

- Data coppia testo in chiaro-testo codificato (p,c)
 - Calcola e memorizza la tavola $D_{k_2}(c)$ per ogni k_2 (richiede 2^k decodifiche e $|k|2^{|k|}$ memoria).
 - per ogni k_1 , verifica se $E_{k_1}(p)$ è nella tavola
 - Ogni verifica positiva dà una coppia di chiavi k_1, k_2
 - Potrebbe essere necessario ripetere più volte (nel caso che per un blocco si trovino più coppie)
- Attacco nel mezzo: utilizzabile per ogni codice iterato riducendo il tempo di attacco banale per un fattore di 2^k codifiche

Codifica tripla



Codifica tripla con CBC:



Codifica dall'esterno (CBC: esterno): si codifica ogni blocco e si concatenano i blocchi codificati **Standard**

- CBC interno (concateno i blocchi per la prima codifica, per la seconda e per la terza) ha alcuni vantaggi

3-DES: Due o tre chiavi

- Alcune volte solo due chiavi sono usate in 3-DES
- La stessa chiave usata all'inizio e alla fine $E_{k1}(D_{k2}(E_{k1}(P)))$ [modalità EDE]
- **Esercizio:** con CBC posso cambiare bit di un blocco del messaggio: lo posso fare con 3-DES interno? 3-DES esterno?

DESX

- progettato da Ron Rivest nel 1984, compete con 3DES rispetto ad attacchi enumerativi (forza bruta)
- Più veloce di 3DES (una sola codifica DES)
- Miglior attacco richiede più di 2^{60} chosen plaintext)
- DESX usa tre chiavi k , k_1 , and k_2
 - $DESX_{k,k_1,k_2}(M) = k_2 \oplus DES_k(k_1 \oplus M)$
 - dimensione chiave $56+64+64=184$ bits

Gruppi e Campi

AES - Advanced Encryption Standard

Gruppi

+ , 0, and -a
sono solo notazioni!

Def (gruppo): Un insieme G con una operazione binaria $+$ (addizione) è un *gruppo* commutativo se

- 1 $\forall a, b \in G, a+b \in G$ (chiusura)
- 2 $\forall a, b, c \in G, (a+b)+c = a+(b+c)$ (associatività)
- 3 $\forall a, b \in G, a+b = b+a$ (commutatività)
- 4 $\exists 0 \in G, \forall a \in G, a+0 = a$ (0 identità)
- 5 $\forall a \in G, \exists -a \in G, a+(-a) = 0$ (esiste inverso)

Gruppi

Z_n interi residuo n ; Z_n^* : residuo mod n con inverso

- insieme interi Z con addizione (0 identità e $-a$ inverso di a)
- insieme Z_n con addizione è un gruppo (0 identità, $-a \pmod n$ inverso di a)
- insieme Z_n con moltiplicazione NON è un gruppo
- insieme di interi con moltiplicazione NON forma un gruppo (esistono inversi solo per 1 e -1)
- insieme $\{1, -1\}$ con moltiplicazione forma un gruppo
- insieme dei razionali con moltiplicazione forma un gruppo
- insieme classi residuo $Z_n^* [a \pmod n]$ con moltiplicazione NON SEMPRE è un gruppo.
 - es. se $n=6$ allora $\{1,2,3,4,5\}$ non è chiuso con moltiplicazione ($2*3 = 0 \pmod 6$)
 - se n è primo allora è un gruppo
- insieme classi residuo $Z_n^* [a \pmod n]$ con moltiplicazione e $\text{MCD}(a,n) = 1$ è un gruppo ($1 \pmod n$ è identità e se $as + nt = 1$ allora $[s \pmod n]$ è l'inverso di $[a \pmod n]$)
 - es. se $n=15$ allora $\{1,2,4,7,8,11,13,14\}$ se $n=5 \{1,2,3,4\}$

Gruppi

Esercizio. Dimostrare che

- Elemento inverso è unico
- Identità è unica

Sottogruppi

Sia $(G, +)$ un gruppo, $(H, +)$ è un sottogruppo di $(G, +)$ se è un gruppo e $H \subseteq G$.

Prop.: Sia $(G, +)$ un gruppo finito e $H \subseteq G$. Se H è chiuso rispetto a $+$, allora $(H, +)$ è un sottogruppo di $(G, +)$

Es.

Gruppi ciclici

Sia a^n pari a $a+\dots+a$ (n volte) a è di ordine n se $a^n = 0$, e, per ogni $m < n$, $a^m \neq 0$

Prop.: Sia G un gruppo e sia a un elemento di ordine n . L'insieme

$\langle a \rangle = \{1, a, \dots, a^{n-1}\}$ è un sottogruppo di G .

a è il generatore di $\langle a \rangle$.

Se G è generato da a , allora G è *ciclico*, e a è un elemento primitivo di G (o anche il generatore di G).

Teorema: per ogni primo p , il gruppo moltiplicativo di Z_p è ciclico

Gruppi ciclici

Z_n interi residuo n ; Z_n^* : residuo mod n con inverso

- insieme Z_n con addizione è un gruppo (0 identità, $-a \bmod n$ inverso di a) generatore è 1 per ogni n
- insieme $\{1, -1\}$ con moltiplicazione forma un gruppo generatore -1
- insieme classi residuo Z_p^* [$a \bmod n$], p primo con moltiplicazione è un gruppo.
 - se $n=7$ allora $\{1,2,3,4,5,6\}$ ordine di 2 è 3 (genera 2,4,1)
3 invece è generatore (genera 3,2,6,4,5,1)

Anelli

Un anello commutativo con identità (o semplicemente un anello) è un insieme R con due operazioni $+$ e \cdot tale che

- $(R,+)$ è un gruppo commutativo con identità 0_R
- la moltiplicazione è commutativa, associativa e distributiva rispetto all'addizione
- esiste un elemento diverso da 0 , sia 1_R t.c. $a \cdot 1_R = a$, per ogni a

Es. l'insieme Z degli interi (o Q dei razionali) con le usuali addizione e moltiplicazione forma un anello

Anelli

$+, \cdot, 0, 1$ e $-a$
sono solo notazioni!

Un insieme F con due operazioni binarie $+$ (addizione) e \cdot (moltiplicazione)

è un *anello* commutativo con identità se

1 $\forall a,b \in F, a+b \in F$

6 $\forall a,b \in F, a \cdot b \in F$

2 $\forall a,b,c \in F, (a+b)+c = a+(b+c)$

7 $\forall a,b,c \in F, (a \cdot b) \cdot c = a \cdot (b \cdot c)$

3 $\forall a,b \in F, a+b = b+a$

8 $\forall a,b \in F, a \cdot b = b \cdot a$

4 $\exists 0 \in F, \forall a \in F, a+0 = a$

9 $\exists 1 \in F, \forall a \in F, a \cdot 1 = a$

5 $\forall a \in F, \exists -a \in F, a+(-a) = 0$

10 $\forall a,b,c \in F, a \cdot (b+c) = a \cdot b + a \cdot c$

Campi

Dato un anello R , un elemento u ha un inverso moltiplicativo se esiste u' t.c. $u \cdot u' = 1$

Un anello commutativo F è un campo se tutti gli elementi hanno un inverso moltiplicativo

Equivalentemente, $(F, +)$ è un gruppo commutativo (additivo) e $(F \setminus \{0\}, \cdot)$ è un gruppo commutativo (moltiplicativo).

Campi

$+, \cdot, 0, 1, -a$
e a^{-1} sono solo notazioni

Def (campo): Un insieme F con due operazioni (addizione) e \cdot (moltiplicazione) è un *campo commutativo con identità* se

- 1 $\forall a, b \in F, a + b \in F$
- 2 $\forall a, b, c \in F, (a + b) + c = a + (b + c)$
- 3 $\forall a, b \in F, a + b = b + a$
- 4 $\exists 0 \in F, \forall a \in F, a + 0 = a$
- 5 $\forall a \in F, \exists -a \in F, a + (-a) = 0$
- 6 $\forall a, b \in F, a \cdot b \in F$
- 7 $\forall a, b, c \in F, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- 8 $\forall a, b \in F, a \cdot b = b \cdot a$
- 9 $\exists 1 \in F, \forall a \in F, a \cdot 1 = a$
- 10 $\forall a, b, c \in F, a \cdot (b + c) = a \cdot b + a \cdot c$
- 11 $\forall a \neq 0 \in F, \exists a^{-1} \in F, a \cdot a^{-1} = 1$ (esiste inverso per tutti gli elementi $\neq 0$)

Esempio

\mathbb{Z}_p denota $\{0,1,\dots,n-1\}$. Si definiscono $+$ e \cdot come addizione e moltiplicazione modulo n , rispettivamente.

Interi modulo n con addizione e moltiplicazione sono un anello commutativo con le seguenti proprietà:

- **Associativa**: $(a+b)+c = a+(b+c) \pmod n$ (vale anche per moltiplicazione)
- **Commutativa** : $a+b = b+a \pmod n$ (vale anche per moltiplicazione)
- **Distributiva** : $(a+b) \cdot c = (a \cdot c) + (b \cdot c) \pmod n$
- **Identità di addizione e moltiplicazione**
 - $a+0 = a \pmod n$ and $a \cdot 1 = a \pmod n$
- **Inverso rispetto all'addizione**
 - $a+(-a) = 0 \pmod n$

Esempi

\mathbb{Z}_n con addizione e moltiplicazione modulo n non è in genere un campo

- $n=15$ NO (se togliamo 0 non otteniamo un gruppo)
- $n=5$ SI (abbiamo visto che $\{1,2,3,4\}$ è un gruppo rispetto alla moltiplicazione)

Teorema \mathbb{Z}_n con addizione e moltiplicazione sono un campo se e solo se p è primo.

Domanda: Esistono altri campi finiti oltre a $(\mathbb{Z}_p, +, \cdot)$?

Campi di Galois $GF(p^k)$

Teorema: Per ogni potenza p^k ($k=1,2,\dots$) di un numero primo esiste un **unico** campo finito con p^k elementi. Questi campi sono denotati $GF(p^k)$.

Non esistono campi finiti con altre cardinalità.



Nota: $GF(p^k)$ e Z_{p^k} **sono diversi!**

Évariste Galois (1811-1832)

Polinomi definiti su campi

Sia $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \dots + a_1 \cdot x + a_0$
un polinomio di grado n in una variabile x su un campo F
(in particolare $a_n, a_{n-1}, \dots, a_1, a_0 \in F$).

Teorema: L'equazione $f(x)=0$ ha al più n soluzioni in F .

Nota: Il teorema non vale su anelli:

Per esempio, in Z_{24} l'equazione $6 \cdot x = 0$
ha sei soluzioni $(0, 4, 8, 12, 16, 20)$.

Polinomi su campi - resti

Siano $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \dots + a_1 \cdot x + a_0$

$g(x) = b_m \cdot x^m + b_{m-1} \cdot x^{m-1} + b_{m-2} \cdot x^{m-2} + \dots + b_1 \cdot x + b_0$

due polinomi su F tale che $m < n$ (o $m=n$).

Teorema: Esiste un unico polinomio $r(x)$ di grado $< m$ su F tale che

$$f(x) = h(x) \cdot g(x) + r(x).$$

Nota: $r(x)$ è il resto di $f(x)$ modulo $g(x)$.

> `rem(4*x^5 + 3*x^2 + 1, x^3+2, x);`

$$1 - 5x^2$$

> `gcd(4*x^5 + 3*x^2 + 1, x^3+2);`

$$1$$

Polinomi su campi finiti

Equazioni polinomiali e fattorizzazione in campi finiti sono in generale diversi nei campi finiti che nei numeri razionali

esempi da sessione XMAPLE :

`factor(x^6-1); # over the rationals`

$$(x-1)(x+1)(x^2+x+1)(x^2-x+1)$$

`Factor(x^6-1) mod 7; # over Z7`

$$(x+1)(x+3)(x+2)(4+x)(x+5)(x+6)$$

`factor(x^4+x^2+x+1); # over the rationals`

$$x^4+x^2+x+1$$

`Factor(x^4+x^2+x+1) mod 2; # over Z2`

$$(x+1)(x^3+x^2+1)$$

Polinomi Irriducibili

Un polinomio è **irriducibile** in $GF(p)$ se non si fattorizza su $GF(p)$. Altrimenti è **riducibile**.

Esempi:

Factor ($x^5+x^4+x^3+x+1$) mod 5;

$(x+2)(x^3+3x+2)(x+4)$

Factor ($x^5+x^4+x^3+x+1$) mod 2;

$x^5+x^4+x^3+x+1$

Lo stesso polinomio è riducibile in Z_5 ma è irriducibile in Z_2 .

Aritmetica di campi finiti $GF(p^k)$

Teorema: Sia $f(x)$ un polinomio **irriducibile** di grado k su Z_p .

Il campo finito $GF(p^k)$ è realizzato dall'insieme di $k-1$ polinomi su Z_p , con addizione e moltiplicazione modulo $f(x)$.

Aritmetica di campi finiti $GF(p^k)$

Per il teorema il campo finito $GF(2^5)$ si realizza come l'insieme dei polinomi di grado 4 su Z_2 , con addizione e moltiplicazione eseguiti modulo il polinomio

irriducibile

$$f(x) = x^5 + x^4 + x^3 + x + 1.$$

I coefficienti di Z_2 sono 0 o 1. Quindi un polinomio di grado k può essere scritto con $k+1$ bits.

Per esempio, $k=4$:

$$x^3 + x + 1 \leftrightarrow (0, 1, 0, 1, 1)$$

$$x^4 + x^3 + x + 1 \leftrightarrow (1, 1, 0, 1, 1)$$

Aritmetica di campi finiti $GF(2^k)$

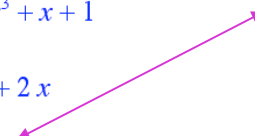
Addizione: bit-a-bit XOR (infatti $1+1=0$)

$$\begin{array}{r} x^3 + x + 1 \quad (0, 1, 0, 1, 1) \\ + \\ x^4 + x^3 + x \quad (1, 1, 0, 1, 0) \\ \hline x^4 \quad + 1 \quad (1, 0, 0, 0, 1) \end{array}$$

Aritmetica di campi finiti $GF(2^k)$

Moltiplicazione: moltiplicazione fra polinomi e quindi calcolo del resto modulo il polinomio $f(x)$ che definisce il campo

```
> g(x) := (x^4+x^3+x+1) * (x^3+x+1);      (1,1,0,1,1) * (0,1,0,1,1)
      g(x) := (x^4+x^3+x+1)(x^3+x+1)      = (1,1,0,0,1)
> f(x) := x^5+x^4+x^3+x+1;
      f(x) := x^5+x^4+x^3+x+1
> rem(g(x), f(x), x);
      1+3x^4+x^3+2x
> % mod 2;
      1+x^4+x^3
```



Per campi finiti di piccola cardinalità il metodo migliore utilizza una tavola per memorizzare i valori

Codici simmetrici a blocchi

out

in

DES

AES

Advanced Encryption Standard (AES)

1997 : NIST (National Inst. of Standard and Tech., USA) Concorso pubblico per AES : nuovo standard a blocchi per uso commerciale. Perché nuovo standard?

- Chiave DES corta, problemi di sicurezza (Matsui: miglior attacco 'lineare' richiede solo 2^{43} coppie di plaintext/ciphertext)
- Esistenza di funzioni trapdoor? (qualcuno pensa che nel progetto di DES permetta al NIST di decodificare anche senza conoscere la chiave)

Advanced Encryption Standard

Requisiti: Cifrario a blocchi con chiavi fra 128 e 256 bit

- Resistente ai tipi di attacco noti
- Semplice, veloce e con codice compatto (implementabile su smart card)
- Senza royalties

Selezione utilizza pubblico esame ed è basata su sicurezza, efficienza implementazione (velocità e memoria richiesta)

Advanced Encryption Standard

SELEZIONE

- **Giugno 1998**: 15 candidati: descrizione algoritmo con analisi efficienza e robustezza rispetto ad attacchi piu' comuni
- **Agosto 1999**: dopo pubblico scrutinio scelta 5 finalisti: Mars(IBM), RC6 (RSA Lab.), Rijndael (Daemen e Rijmen), Serpent (Anderson et al.), Twofish (Schneier et al.)
- **Ottobre 2000**: RIJNDAEL è stato scelto: pubblico esame e eventuale revisione
- **2001** : RIJNDAEL è proposto come standard AES
- **2005**: lo standard FIPS di DES è ritirato

Advanced Encryption Standard

AES usa chiavi di 128, 192 o 256 bit

128 bit: 10 (rounds) iterazioni

192 (256) bit: 12 (14) iterazioni

codifica blocchi di 128 bit ciascuno

- **Resiste a tutti gli attacchi noti**
- **Veloce con codice compatto**
- **Semplice**

Rounds in AES: chiavi da 128 bit

128 bits AES usa 10 rounds

- La chiave segreta è espansa da 128 bits in 10 **chiavi di round**, K_0, K_1, \dots di 128 bits ciascuna
- il blocco da codificare è diviso in byte (ottetti) organizzati a matrice
- Inizialmente lo stato è il blocco da codificare in XOR con chiave K_0
- Ogni round cambia lo stato e poi XOR con la **chiave di round K_i**
- Operazioni usate nel round: Sostituzione di blocchi di un byte, permutazione dei byte (righe e colonne della matrice)

AES usa la chiave solo XOR con lo stato e usa la stessa sequenza di chiavi su ogni blocco: Attaccabile come in one-time pad quando si usa la stessa chiave?

No: Ogni round trasforma il blocco e complica le cose.

Complessivamente sembra **impossibile invertire** senza la chiave segreta senza enumerazione (facile se si conosce la chiave)

Definizione AES: blocco

- Lunghezza dei blocchi di input & output: 128 bits.
- Stato: 128 bits, ordinati in una matrice 4×4 di byte (ottetti)

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$

Ogni byte è un elemento di $GF(2^8)$

Input/Output: $A_{0,0}, A_{1,0}, A_{2,0}, A_{3,0}, A_{0,1}, \dots$

Definizione AES: chiave

Lunghezza della chiave: 128, 192, 256 bits.

Layout della chiave $n = 128, 192, 256$ bits, organizzati in una matrice di byte di dimensione 4-per- $n/32$

Es. chiave da 192 bit

$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$	$K_{0,4}$	$K_{0,5}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	$K_{1,4}$	$K_{1,5}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	$K_{2,4}$	$K_{2,5}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	$K_{3,4}$	$K_{3,5}$

layout iniziale: $K_{0,0}, K_{1,0}, K_{2,0}, K_{3,0}, K_{0,1}, \dots$

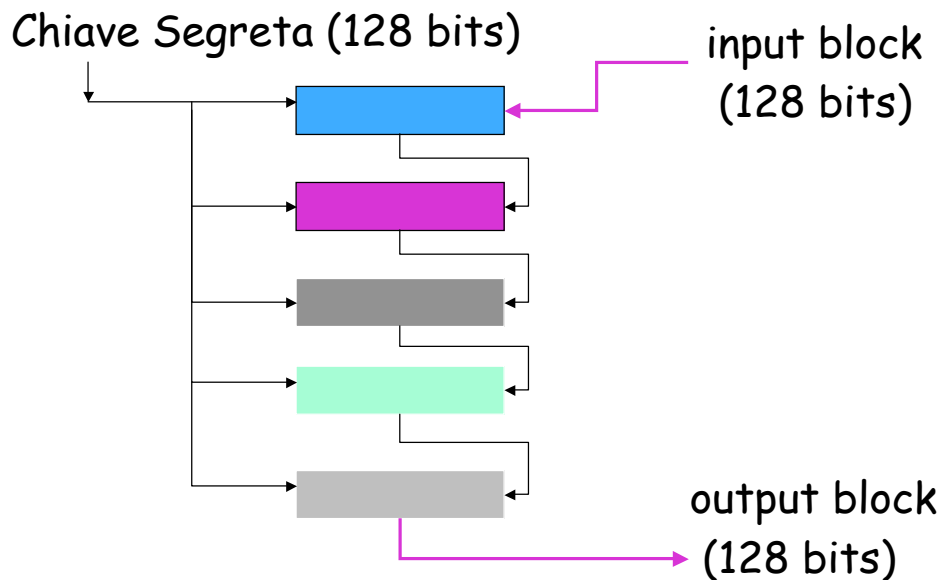
Definizione AES:128 bit

Codice alto livello

AES(Blocco,Chiave)

- Stato=Blocco (organizzato a matrice di ottetti)
- KeyExpansion(Key,ExpandKey)
- XOR(Stato,ExpandKey[0])
- For (i=1; i<R; i++)
 - Round(State) (modifica con sostituzioni e permutazioni);
 - XOR(Stato,ExpandKey[i]);

Codifica eseguita in rounds



Specifica AES: un Round

Trasforma lo stato applicando:

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$

1. Sostituzione.
2. Shift righe
3. Mix colonne
4. XOR fra stato
e la chiave di round

Sostituzione (S-Box)

Sostituzione definita da una matrice (S-Box)
per ogni i,j sostituisce $A_{i,j}$ con elemento ij della matrice
(con $A_{i,j}^{-1}$ inverso di $A_{i,j}$ in $GF(2^8)$, $GF=$ Campo di Galois)
e poi moltiplica per polinomio $x^4 + x^3 + x^2 + x + 1$
infine somma polinomio $x^6 + x^5 + x + 1$
(tutte le operazioni sono fatte in Z_2 modulo polinomio x^8+1)

Se $i=j=0$ allora $A_{i,j}=0$, non si modifica $A_{i,j}$

Chiaramente la sostituzione è **invertibile**.

Nota Bene: operazione non lineare

Shift ciclico delle righe

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,3}$	$A_{1,0}$	$A_{1,1}$	$A_{1,2}$
$A_{2,2}$	$A_{2,3}$	$A_{2,0}$	$A_{2,1}$
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,0}$

no shift

shift 1 posizione

shift 2 posizione

shift 3 posizione

Chiaramente lo shift è **invertibile**.

Mescolamento delle Colonne

Ogni stato viene considerato come un polinomio $GF(2^8)$ che viene moltiplicato con il polinomio $03x^3 + 01x^2 + 01x + 02 \pmod{x^4 + 1}$

Nota:

operazione invertibile moltiplicando per

$Inv = 0Bx^3 + 0Dx^2 + 09x + 0E$

Operazione può essere implementata con tabella di 256 righe in cui ciascuna riga contiene 128 bit; sfruttando proprietà algebriche si arriva a tabella con 256 righe e 32 bit

Implementazione

- S-Box, shift righe, Mescolamento colonne ciascuna richiede una tabella con 256 righe e 32 bit per riga
- Analogamente XOR con chiave

Nota:

- le operazioni sono fatte in sequenza e non dipendono dalla chiave: basta una sola tabella + uso di operazioni di shift
- XOR con chiave può essere fatto memorizzando la chiave di round e poi facendo direttamente XOR quando si calcola corrispondente valore

Espansione della chiave

- Genera una chiave "diversa" per ogni round
- Si utilizza una matrice 4×4 di (su $GF(2^8)$) per round
- Si basa su una trasformazione non lineare della chiave originale
- Dettagli disponibili su:

<http://en.wikipedia.org/wiki/AES>

<http://csrc.nist.gov/CryptoToolkit/aes/>

The Design of Rijndael, Joan Daemen and Vincent Rijmen,
Springer Verlag

Rompere AES

- efficienza (anche su elaboratori piccole dimensioni):
 - S-box: ricerca in tabella (piccole dimensioni)
 - mescolamento righe: shift
 - permutazione colonne: tabella,
 - XOR: veloce
- sicurezza:
 - Rompere 1 o 2 rounds facile
 - Non si sa come farlo per 5 rounds; rompere 10 rounds AES in modo efficiente (es. 1 anno con hardware esistente o in meno di 2^{128} operazioni, ad es. 2^{100}) e' **impossibile** (per il momento) !

Esercizi

1. Valutare propagazione errori in CBC e OFB:
 - mostrare come l'avversario possa modificare un blocco arbitrariamente a prezzo di una modifica parziale del resto del messaggio
 - mostrare la non sicurezza di questo e discutere possibili tecniche per evitare questa situazione
2. CBC e OFB utilizzano un valore iniziale (IV) che deve essere noto sia al mittente che al ricevente
 - mostrare come se il valore iniziale è mandato in chiaro un avversario possa modificare parte del messaggio (quindi o IV è fissato a priori oppure deve essere inviato in crittato e inviato prima del resto del messaggio)
3. Rompere OFB se si usano la stessa chiave e il IV più volte