Cryptography and Network/Information Security

Advanced Encryption Standard

Outline

- Processo di selezione per AES
- Dettagli del cipher Rijndael-AES
- Varii passi dei varii round
- Key expansion
- Implementazione

Introduzione

- Trattati i possibili attacchi al DES
  - Attacco a forza bruta
  - Criptanalisi lineare
  - Criptanalisi differenziale
- Apparsa esigenza di cipher con maggiore robustezza
- Proposta del Triple-DES, però lento e con blocchi piccoli
- Nel 1997 il NIST emette un bando per un nuovo cipher con specifiche migliori del DES
- Nel giugno 1998 15 proposte superano una preselezione
- Nel 1999 sono selezionate 5 delle 15
- In ottobre 2000 è selezionato come AES il Rijndael
- In novembre 2001 emesso come standard (FIPS PUB 197)

Richieste Per AES

- Cifrario simmetrico a blocco con chiave privata
- Blocco dati da 128 bit, chiavi da 128/192/256 bit
- Più veloce e più robusto del Triple-DES
- Alta efficienza computazionale
- Vita attiva di 20÷30 anni (più uso di archivio)
- Algoritmo usabile in modo libero (no royalties)
- Richiesta di specifica completa e dei dettagli
- Implementazioni in C ed in Java
- Il NIST pubblicherà tutto quanto presentato e declassificherà le analisi

Criteri di selezione

I criteri di selezione cambiarono lungo la selezione stessa

- Inizialmente:
  - Sicurezza – Come sforzo per una criptoanalisi fattibile
  - Costo – Come efficienza computazionale
  - Caratteristiche di implementazione degli algoritmi
- Alla fine:
  - Sicurezza in senso ampio (analisi pubbliche)
  - Facilità di realizzazione hardware e software
  - Resistenza agli attacchi alla implementazione
  - Flessibilità

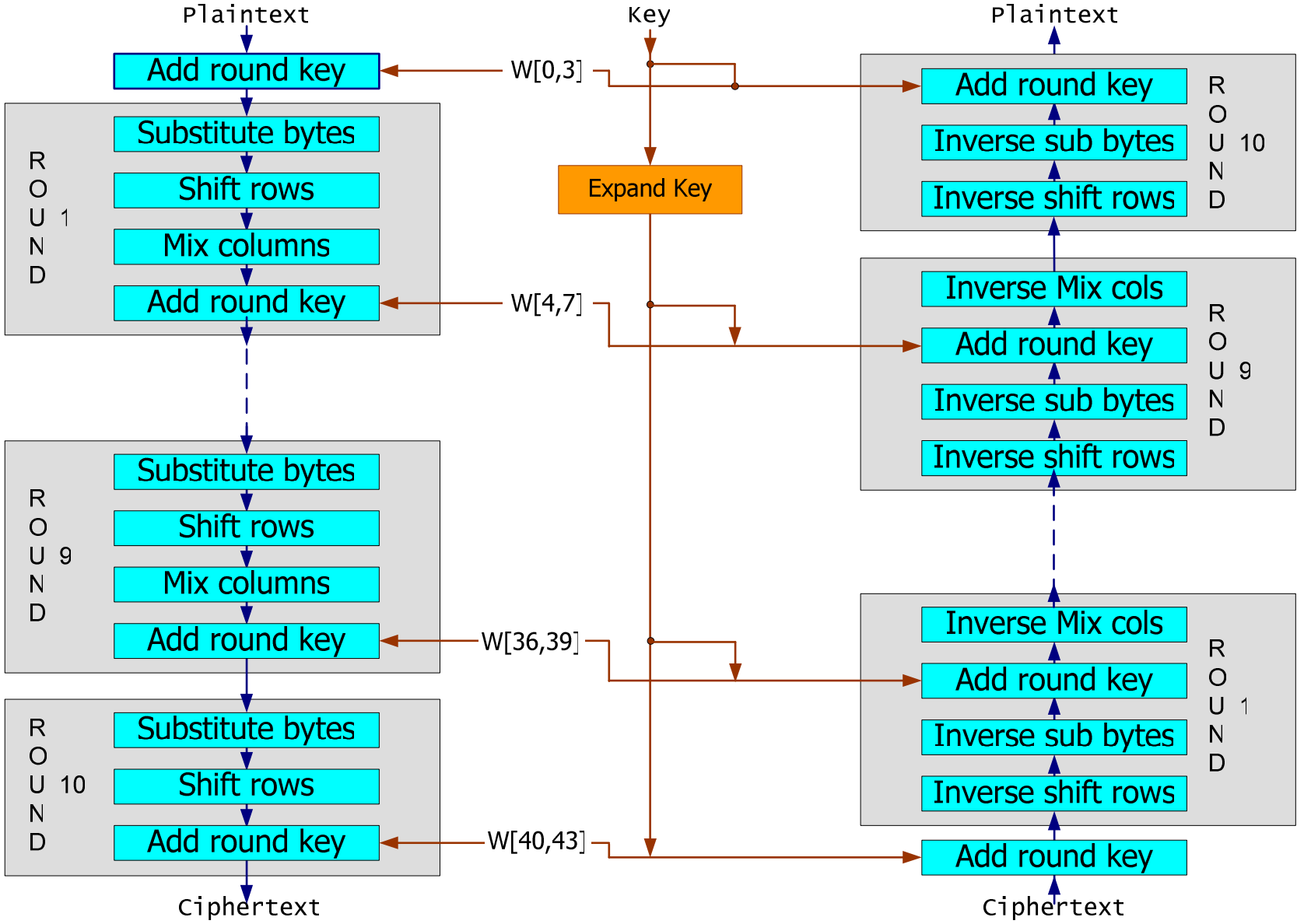
Prima selezione

- In agosto 1998 5 proposte superano la 1° selezione
  - MARS (IBM) - complesso, veloce, alto margine di sicurezza
  - RC6 (USA) – molto semplice e veloce, basso margine di sicurezza
  - Twofish (USA) - complesso, molto veloce, alto margine di sicurezza
  - Serpent (Europa) - lento, chiaro, altissimo margine di sicurezza
  - Rijndael (Belgio) - chiaro, veloce, buon margine di sicurezza
- Successiva fase di analisi e commento
- Si palesa una contrapposizione tra algoritmi con:
  - Pochi stadii complessi o molti stadii semplici
  - Ridefinizioni di algoritmi precedenti o algoritmi del tutto nuovi

Cifrario Rijndael

- Progettato in Belgio da V. Rijmen e J. Daemen
- Blocchi di dati da 128 bit e chiavi da 128/192/256 bit
- Cifrario non Feistel ma iterativo
  - Elabora i dati come blocchi (State) di 4 words da 4 byte
  - In ogni stadio opera sull'intero blocco di dati
- Obiettivi del progetto
  - Resistenza a tutti gli attacchi conosciuti
  - Velocità e compattezza di codice sulle CPU più diffuse
  - Semplicità concettuale

Rijndael (chiave 128 bit)

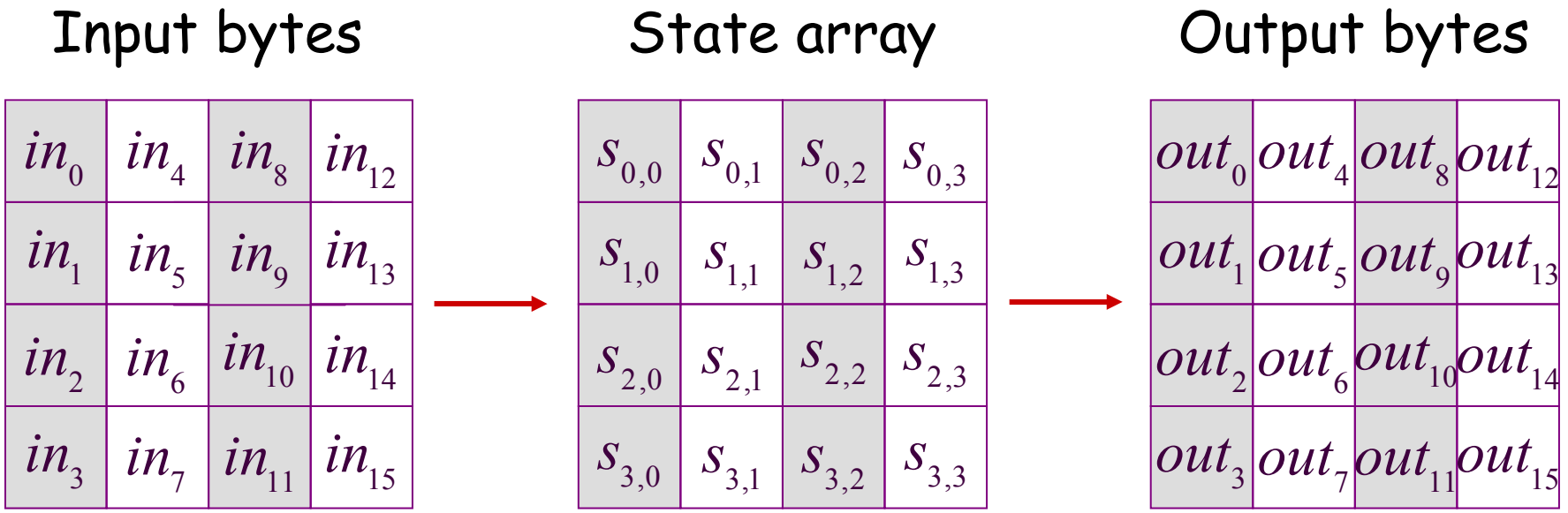


Parametri di AES

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

State array

Operazioni dell'algoritmo effettuate su un array bidimensionale indicato come lo State



Rijndael Stages

4 stadii differenti

- Substitute Bytes
- Shift Rows
- Mix Columns
- AddRoundKey

Substitute Bytes Transformation

Forward and Inverse Substitute byte transformation

Parliamo della Forward Transformation

Uno State formato da 16 bytes

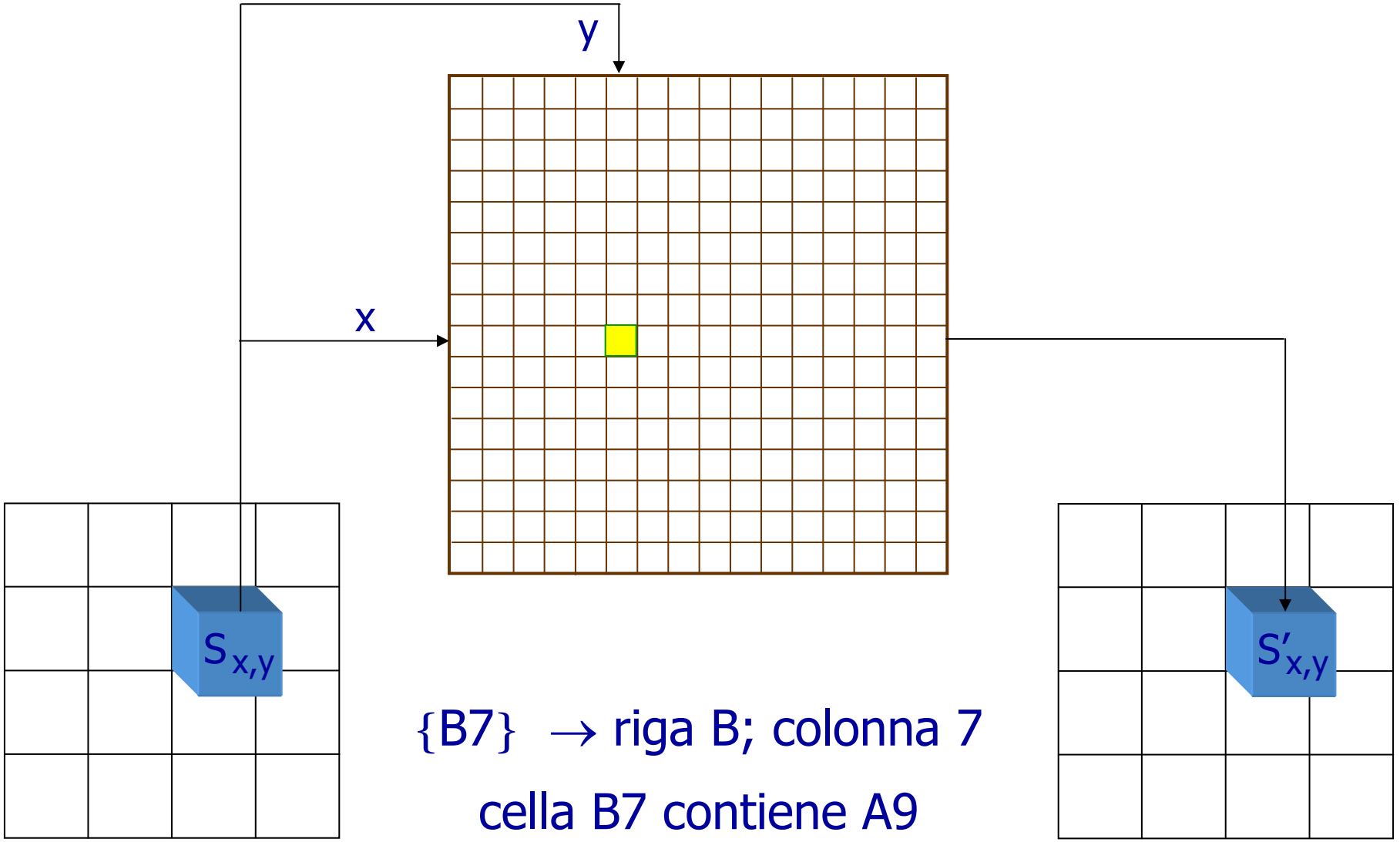
Matrice di 16 x 16 = 256 possibili valori di un byte

I 4 bit più significativi del byte sono usati come indice di riga mentre i 4 meno significativi lo sono come indice di colonna

Questi due indici permettono di selezionare un valore in uscita

La cella selezionata contiene il valore da sostituire

Substitute Bytes Transformation



{B7} → riga B; colonna 7

cella B7 contiene A9

Costruzione S-box

1) Inizializzazione dell'S-box con valori di byte ascendenti

```
00 01 02 03 04 . . . . 0F
10 11 12 13 14 . . . . 1F
. . . . . . . . . . . . .
F0 F1 F2 F3 F4 . . . . FF
```

2) Mappaggio di ciascun byte dell'S-box nel suo inverso moltiplicativo nel campo $GF(2^8)$

3) Definito ciascun byte come formato dagli 8 bit $b_7, b_6, b_5, \ldots, b_0$ si applica a ciascun bit di ogni byte la trasformazione affine su $GF(2)$

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

ove c_i è il bit i-esimo di un byte c = {63}

Inverse Substitute byte transformation

Identica a quella forward ma usando l'S-box inverso

1) Applicazione della trasformazione

$$b_i^{'} = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

ove d_i è il bit i-esimo di un byte d = {05}

2) Mappaggio di ciascun byte dell'S-box nel suo inverso moltiplicativo nel campo $GF(2^8)$

S-box Rijndael progettato perché l'output non possa essere descritto come una semplice funzione matematica dell'input e per ottenere una bassa correlazione tra bit di ingresso e di uscita

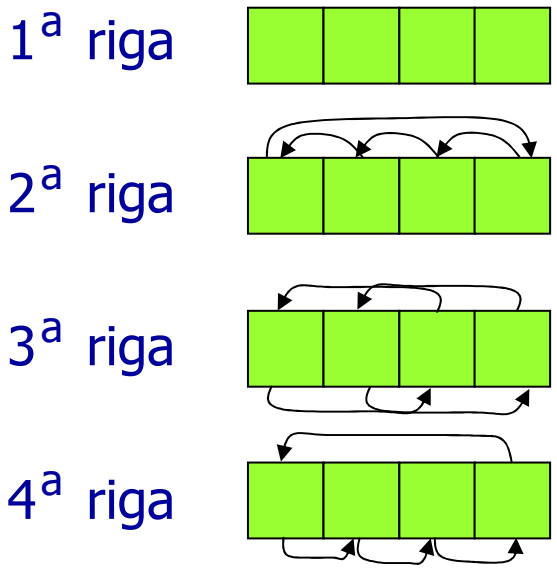
Shift Rows Transformation

Ciascuna riga dello State è shiftata ciclicamente un definito numero di volte

- 1^{a} riga non è cambiata
- 2^{a} riga è shiftata circolarmente di 1 byte a sinistra
- 3^{a} riga è shiftata circolarmente di 2 byte a sinistra
- 4^{a} riga è shiftata circolarmente di 3 byte a sinistra

La inverse transformation opera gli stessi shift, ma verso destra

Shift Row Transformation (2)



Questa operazione serve a distribuire i quattro bytes di una colonna sopra le quattro colonne differenti

Mix Columns Transformation

Anche qui una forward ed una inverse transformation

Forward transformation

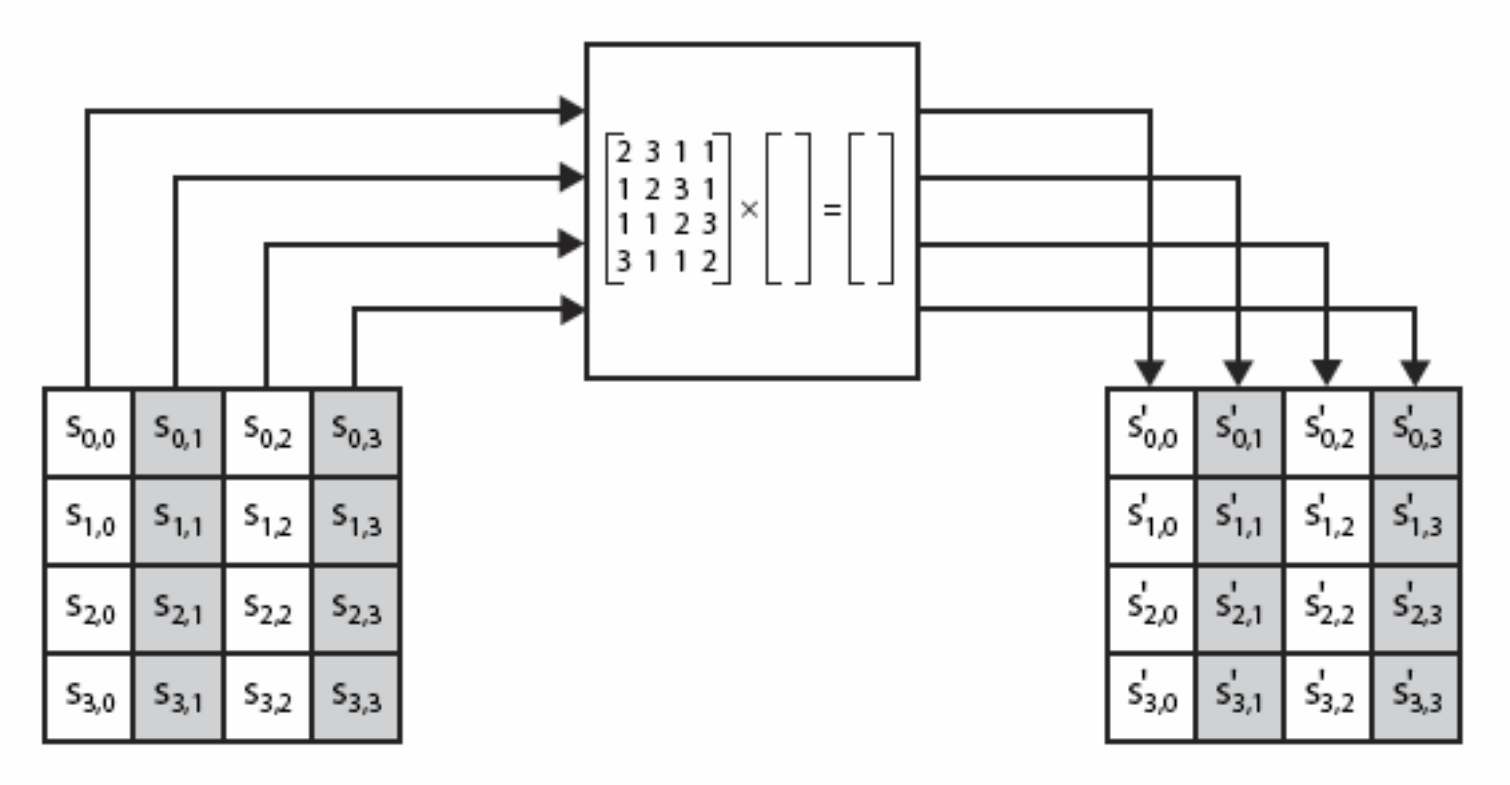
Opera individualmente su ciascuna colonna

Ogni byte è mappato in un valore funzione di tutti i bytes della colonna

La trasformazione può essere espressa come un prodotto matriciale in $GF(2^8)$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Mix Columns Transformation (2)



Mix Columns Transformation (3)

Inverse transformation

La stessa procedura ma con la matrice inversa

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

Trasformazione realizzabile anche come un'operazione tra polinomi

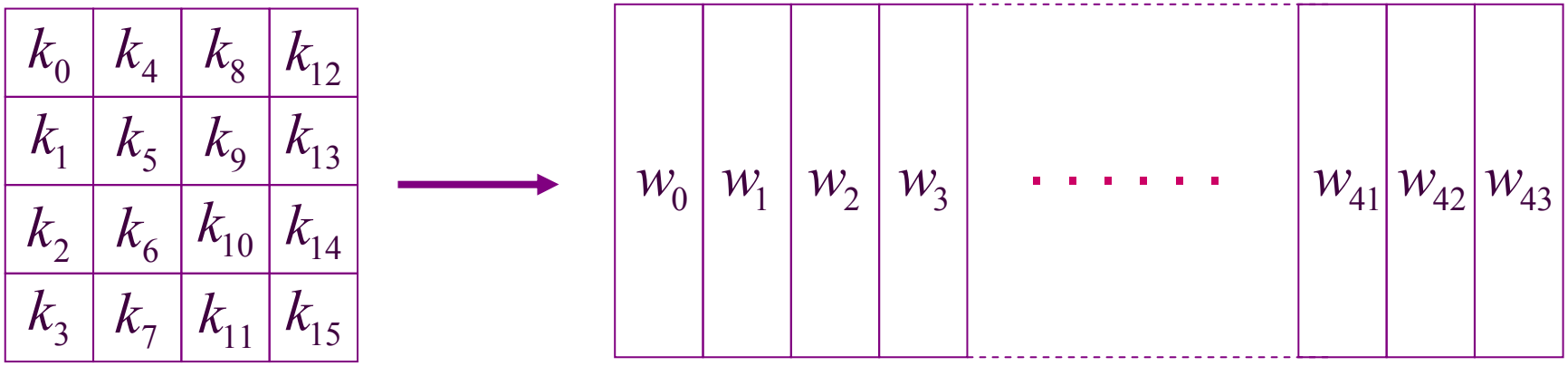
Con questa trasformazione si ottiene un buon mixing tra i bytes delle diverse colonne. Dopo pochi stadii tutti i bit in output dipendono da tutti i bit in input.

AddRoundKey Transformation

- L'operazione AddRoundKey consiste in un bitwise XOR tra i 128 bits dello State e i 128 bits della round key
- Nel processo di encryption si usano 11 round key di 4 words
- In totale si usano 44 words da 4 bytes = 176 bytes = 1408 bits
- AES usa una chiave di 128 bit (4 words da 4 bytes)
- 1408 bits ottenuti tramite il processo di key expansion

Key expansion

128 bits key $\rightarrow$ 16 bytes $\rightarrow$ 4 words



44 round key da 4 words

Key expansion (2)

Criteri di progetto per l'algoritmo di key expansion

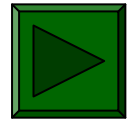
- Semplicità della descrizione
- Non-linearità - Proibisce la piena determinazione delle differenze tra le chiavi di round dalle differenze tra le chivi di cipher
- Diffusione - Ciascun bit della chiave influenza molti bit delle round key
- Costante di round - Elimina la simmetria o le simiglianze tra i modi di generazione delle chiavi di round
- Trasformazione invertibile
- Veloce anche con processori poco potenti

Key expansion (3)

Le words w_i della expanded key sono calcolate nell'ordine dell'indice

Le prime 4 words w_i sono le prime 4 words della key da 128 bits

Le sucessive words sono l'XOR tra la word indietro di 4 posti e un'altra word che, se (i mod 4) ≠ 0 è la word indietro di 1 posto calcolata appena prima mentre, se (i mod 4) = 0 è l'output di una funzione complessa g applicata alla word indietro di 1 posto



Key expansion (4)

$$g = SubWord\left(RotWord(w_{i-1})\right) \oplus Rcon(i/4)$$

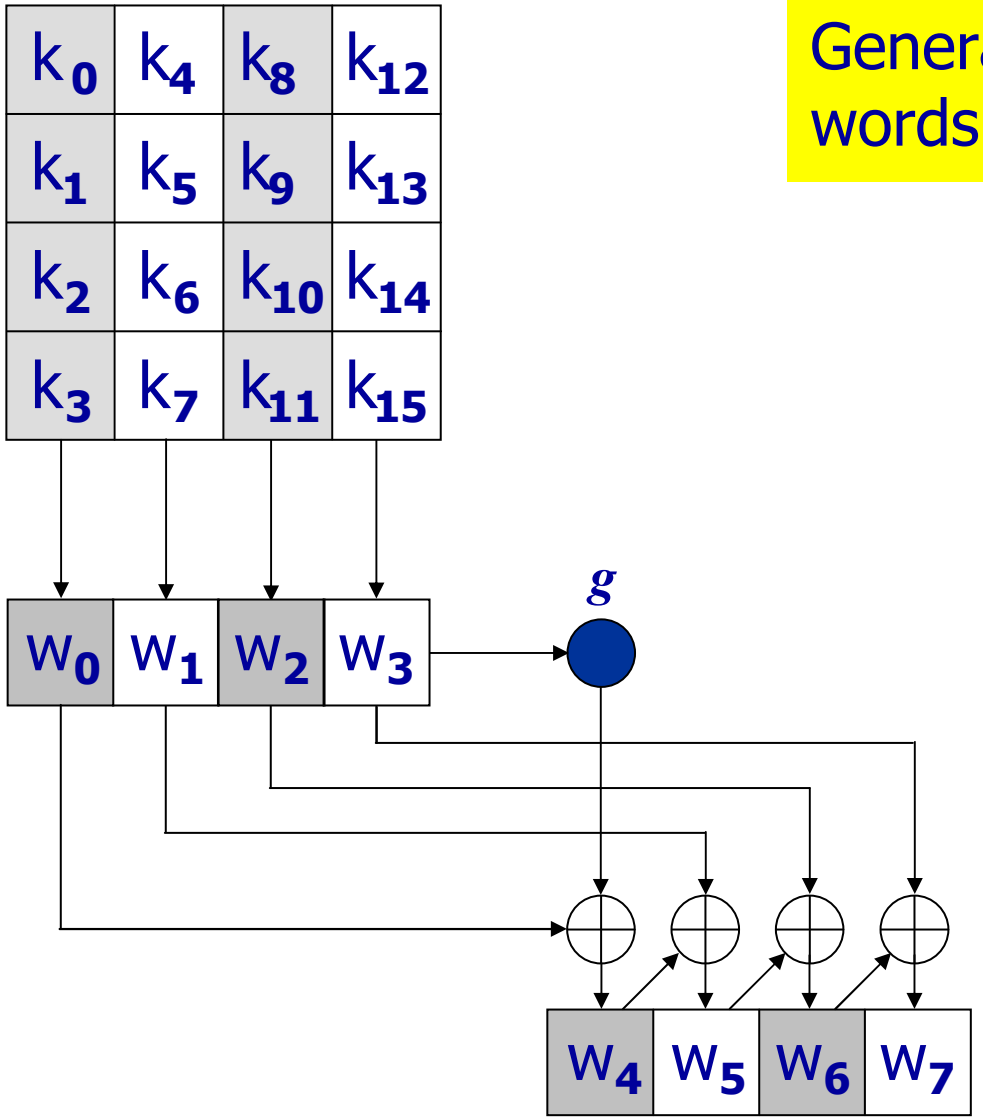
SubWord effettua una sostituzione di byte usando la S-box

RotWord compie uno shift circolare a sinistra di una word

Rcon(j) è una costante di round con i 3 bytes a sinistra nulli

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

Key expansion (5)

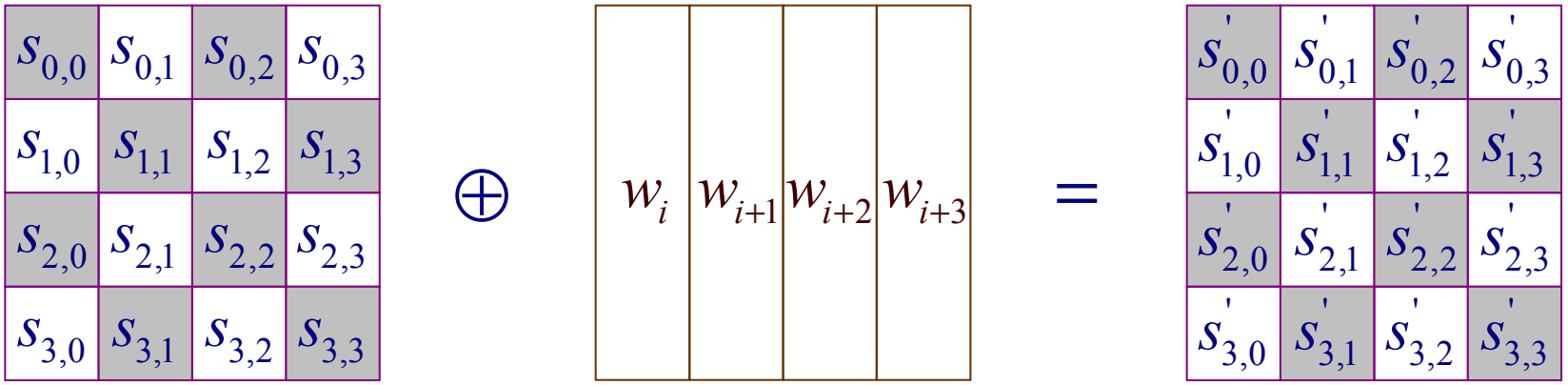


Generazione delle prime otto words della expanded key

AddRoundKey Transformation (2)

Operazioni tra i 4 bytes di una colonna dello State

e una word della round key



Equivalent Inverse Cipher

Il decryption cipher AES non è identico all'encryption cipher perché i diversi passi si susseguono all'inverso

Necessità quindi di due moduli software differenti

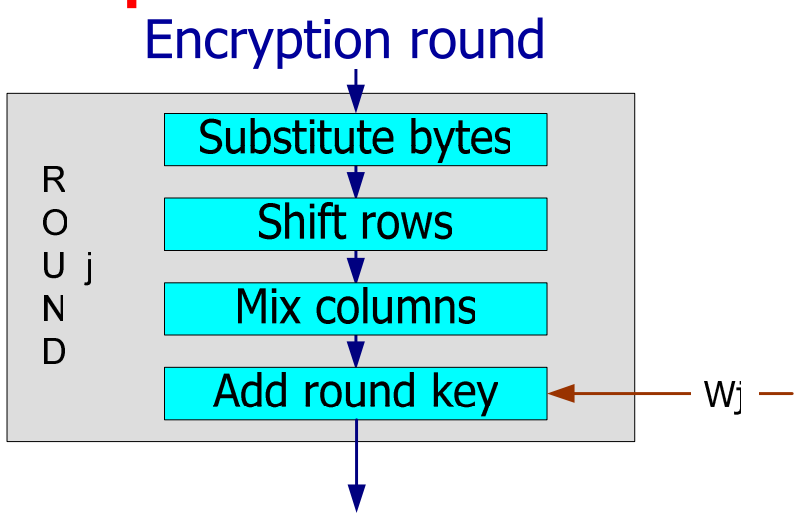
Possibile però definire un algoritmo di decryption equivalente con la stessa sequenza di operazioni di quello di encryption

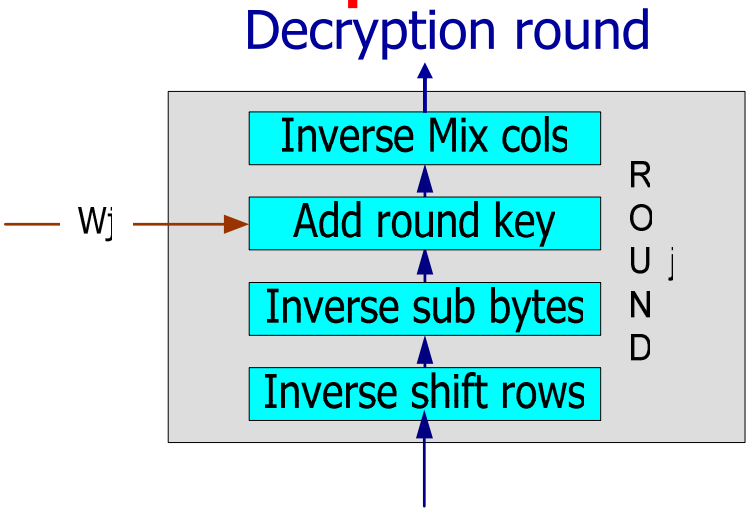
Questo richiede:

Sostituire le trasformazioni con le loro inverse

Variazione nella schedulazione delle chiavi

Equivalent Inverse Cipher (2)

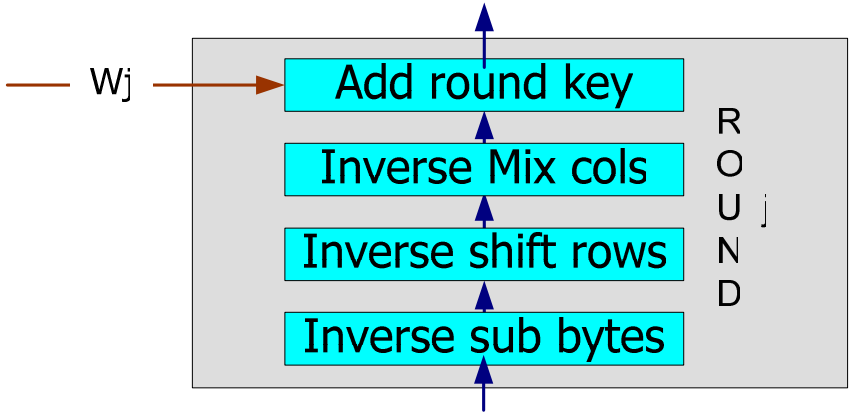




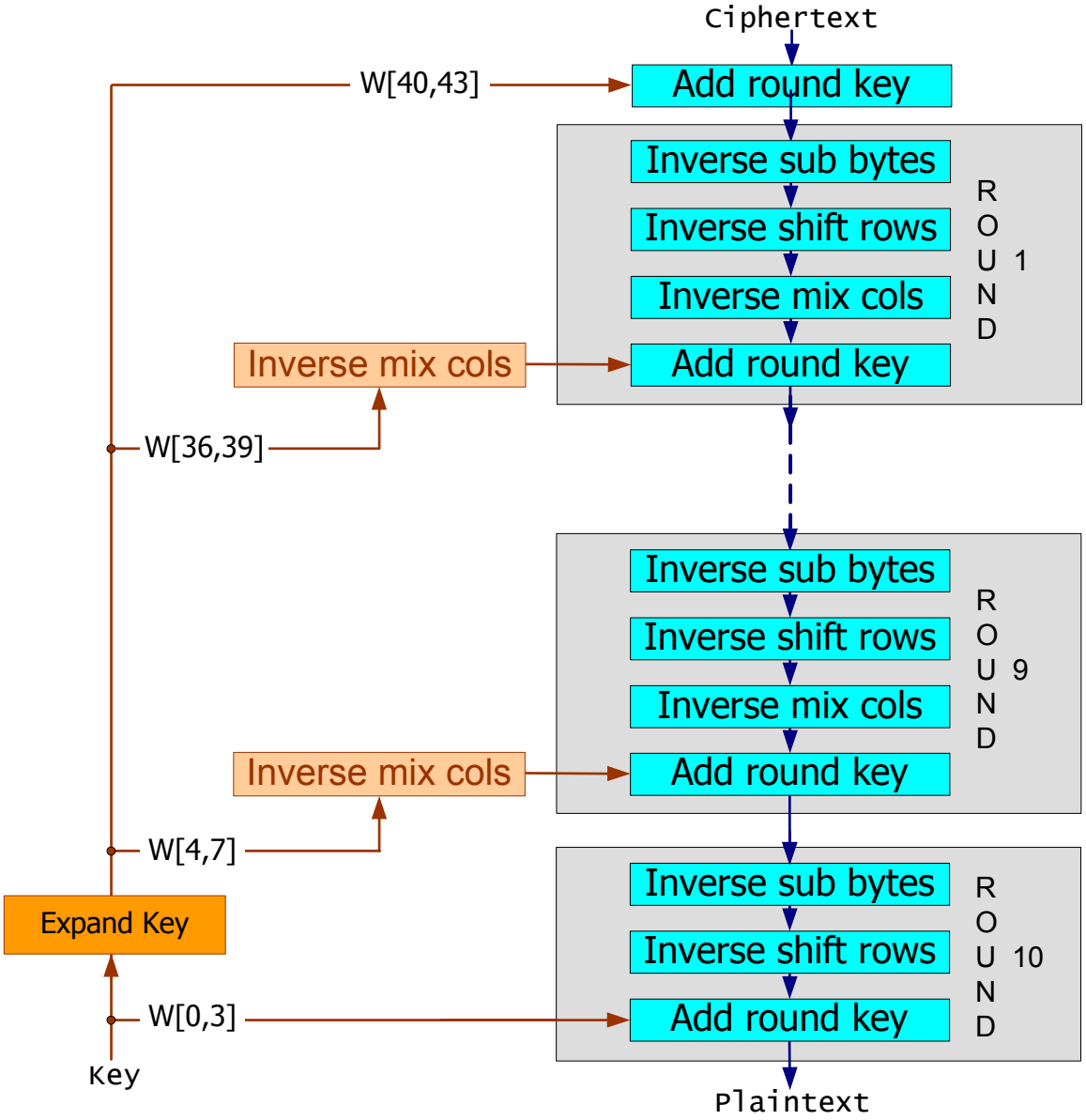
Bisogna scambiare l'ordine tra le prime 2 operazioni e tra le seconde 2

Inverse shift rows e Inverse sub bytes possono essere scambiate tra loro

Per scambiare tra loro Inverse Mix Cols e Add round key è necessario fare applicare alla round key una operazione di Inverse mix columns



Equivalent Inverse Cipher (3)



Implementazione

Può essere implementato in modo efficiente su CPU a 8 bit (smartcard)

L'operazione AddRoundKey è un bitwise XOR

L'operazione Shift Rows è un semplice shift a livello byte

L'operazione Substitute Bytes lavora con una tavola di 256 voci

L'operazione Mix Columns chiede una moltiplicazione di matrici in $GF(2^8)$ ma può essere semplificata usando tabelle e XOR a livello byte

Implementazione (2)

Può essere implementato efficientemente su CPU a 32 bit

Ridefinizione dei passi per usare words a 32 bit

Possibile precalcolare 4 tabelle di 256 words

> Dopo ciascuna colonna in ciascun round può essere calcolata usando 4 table lookups + 4 XORs
>
> Con un costo di 4 Kb per conservare le tabelle

I progettisti ritengono che questa efficienza di implementazione sia stato uno degli elementi chiave per essere scelto come AES

Sicurezza di AES

Ad oggi (2006) si ritiene possibile condurre contro AES soltanto attacchi del tipo side channel

Attacco side channel è un attacco basato su informazioni acquisite tramite l'implementazione fisica del sistema criptografico come potrebbero essere quelle riguardanti il tempo di encryption, la potenza assorbita, analisi delle radio-onde emesse, etc