Advanced Encryption Standard-II

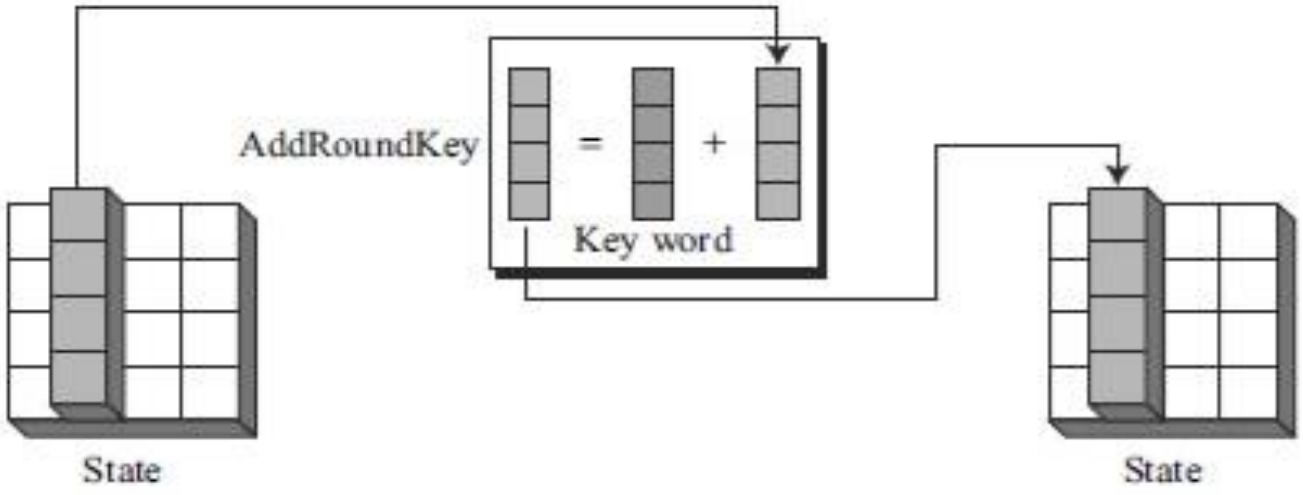
Sachin Tripathi

IIT(ISM), Dhanbad

Key Addition Layer

- The two inputs to the Key Addition layer are the current 16-byte state matrix and a subkey which also consists of 16 bytes (128 bits).
- The two inputs are combined through a bitwise XOR operation.
- Note that the XOR operation is equal to addition in the Galois field GF(2).

Add Round



Algorithm

Pseudocode for AddRoundKey transformation

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_4 round + c
}
```

Key Expansion

- To create round key for each round, AES uses a key-expansion process. If the number of rounds is N_r , the key-expansion routine creates $N_r + 1$, 128-bit round keys from one single 128-bit cipher key.
- The first round key is used for pre-round transformation (AddRoundKey)
- Remaining round keys are used for the last transformation (AddRoundKey) at the end of each round.
- The key-expansion routine creates round keys word by word, where a word is an array of four bytes. The routine creates $4 \times (N_r + 1)$ words that are called w_0, w_1, w_2, …, $w_{4(Nr+1)-1}$

  (In other words, in the AES-128 version (10 rounds), there are 44 words)

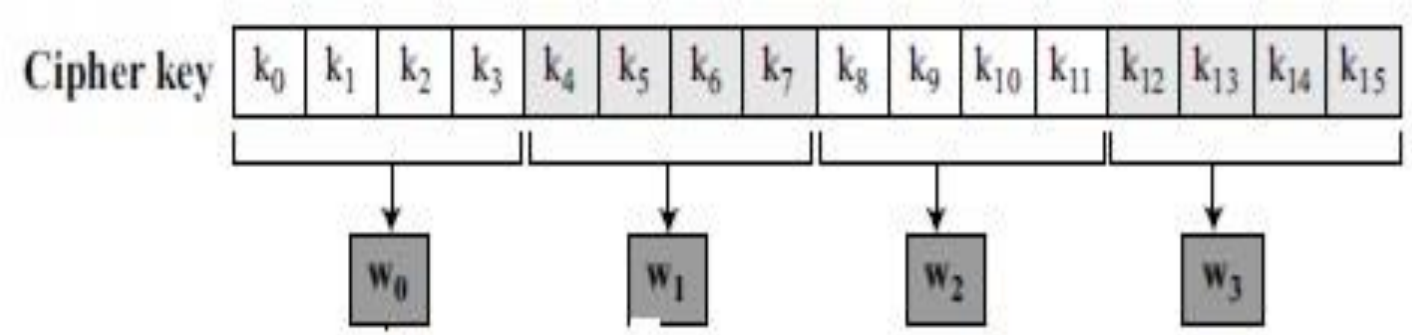
Words for each Round

Round	*Words*			
Pre-round	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7
2	w_8	w_9	w_{10}	w_{11}
. . .	. . .			
N_r	w_{4N_r}	w_{4N_r+1}	w_{4N_r+2}	w_{4N_r+3}

Key Expansion Process

The first four words (w_0, w_1, w_2, w) are made from the cipher key.

The cipher key is thought of as an array of 16 bytes (k_0 to k_{15}).



The rest of the words (w_i for i = 4 to 43) are made as follows:

If (i mod 4) ≠ 0, $w_i = w_{i-1} \oplus w_{i-4}$

If (i mod 4) = 0 , $w_i = t \oplus w_{i-4}$

$$t = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}_{i/4}$$

- **RotWord**

The RotWord (rotate word) routine is similar to the ShiftRows transformation, but it is applied to only one row. The routine takes a word as an array of four bytes and shifts each byte to the left with wrapping.

- **SubWord**

The SubWord (substitute word) routine is similar to the SubBytes transformation, but it is applied only to four bytes. The routine takes each byte in the word and substitutes another byte for it.

Round Constant

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\mathbf{01}\ 00\ 00\ 00)_{16}$	6	$(\mathbf{20}\ 00\ 00\ 00)_{16}$
2	$(\mathbf{02}\ 00\ 00\ 00)_{16}$	7	$(\mathbf{40}\ 00\ 00\ 00)_{16}$
3	$(\mathbf{04}\ 00\ 00\ 00)_{16}$	8	$(\mathbf{80}\ 00\ 00\ 00)_{16}$
4	$(\mathbf{08}\ 00\ 00\ 00)_{16}$	9	$(\mathbf{1B}\ 00\ 00\ 00)_{16}$
5	$(\mathbf{10}\ 00\ 00\ 00)_{16}$	10	$(\mathbf{36}\ 00\ 00\ 00)_{16}$

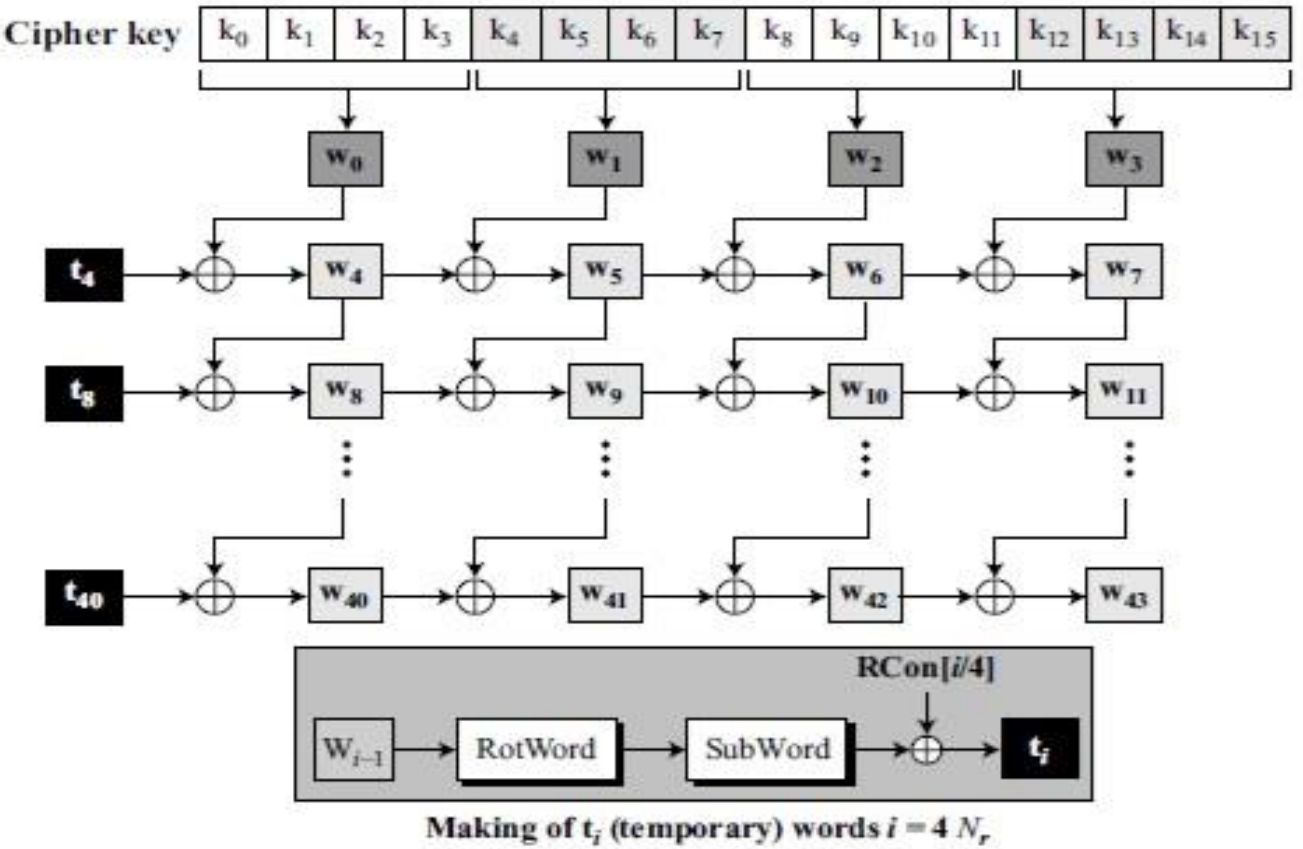
The key-expansion routine can either use the above table when calculating the words

RC_1	$\rightarrow x^{1-1}$	$= x^0$	mod *prime*	$= 1$	$\rightarrow 00000001$	$\rightarrow 01_{16}$
RC_2	$\rightarrow x^{2-1}$	$= x^1$	mod *prime*	$= x$	$\rightarrow 00000010$	$\rightarrow 02_{16}$
RC_3	$\rightarrow x^{3-1}$	$= x^2$	mod *prime*	$= x^2$	$\rightarrow 00000100$	$\rightarrow 04_{16}$
RC_4	$\rightarrow x^{4-1}$	$= x^3$	mod *prime*	$= x^3$	$\rightarrow 00001000$	$\rightarrow 08_{16}$
RC_5	$\rightarrow x^{5-1}$	$= x^4$	mod *prime*	$= x^4$	$\rightarrow 00010000$	$\rightarrow 10_{16}$
RC_6	$\rightarrow x^{6-1}$	$= x^5$	mod *prime*	$= x^5$	$\rightarrow 00100000$	$\rightarrow 20_{16}$
RC_7	$\rightarrow x^{7-1}$	$= x^6$	mod *prime*	$= x^6$	$\rightarrow 01000000$	$\rightarrow 40_{16}$
RC_8	$\rightarrow x^{8-1}$	$= x^7$	mod *prime*	$= x^7$	$\rightarrow 10000000$	$\rightarrow 80_{16}$
RC_9	$\rightarrow x^{9-1}$	$= x^8$	mod *prime*	$= x^4 + x^3 + x + 1$	$\rightarrow 00011011$	$\rightarrow 1B_{16}$
RC_{10}	$\rightarrow x^{10-1}$	$= x^9$	mod *prime*	$= x^5 + x^4 + x^2 + x$	$\rightarrow 00110110$	$\rightarrow 36_{16}$

Use the GF(2^8) field to calculate the leftmost byte dynamically.

The leftmost byte, which is called RC_i is actually x^{i-1}, where i is the round number. AES uses the irreducible polynomial ($x^8 + x^4 + x^3 + x + 1$).

Key Expansion in AES-128



Algorithm

```
KeyExpansion ([key0 to key15], [w0 to w43])
{
    for (i = 0 to 3)
        wi ← key4i + key4i+1 + key4i+2 + key4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    wi ← wi–1 + wi–4
        else
        {
            t ← SubWord (RotWord (wi–1)) ⊕ RCon i/4        // t is a temporary word
            wi ← t + wi–4
        }
    }
}
```

Key Expansion Example

Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

Illustration

RotWord (13AA5487) = AA548713 → **SubWord** (AA548713) = AC20177D

$\mathbf{t} = \text{AC20177D} \oplus \mathbf{RCon_1} = \text{AC20 17 7D} \oplus 01000000_{16} = \text{AD20177D}$

The two sets of round keys can be created from two cipher keys that are different only in one bit.

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC AA 34 C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC A**B** 34 C2 BB 77 23

Comparing two sets of round keys

R.	*Round keys for set 1*	*Round keys for set 2*	*B. D.*
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9EDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83EDAB0 FDD00348 A0A90064 D2EBF651	52

Key Expansion in AES -192

In AES-192, the words are generated in groups of six instead of four.

The cipher key creates the first six words (w_0 to w_5).

If i mod 6 $\neq$ 0, $w_i \leftarrow w_{i-1} + w_{i-6}$; otherwise,

$$w_i \leftarrow t + w_{i-6}$$

Key Expansion in AES -256

In AES-256, the words are generated in groups of eight instead of four.

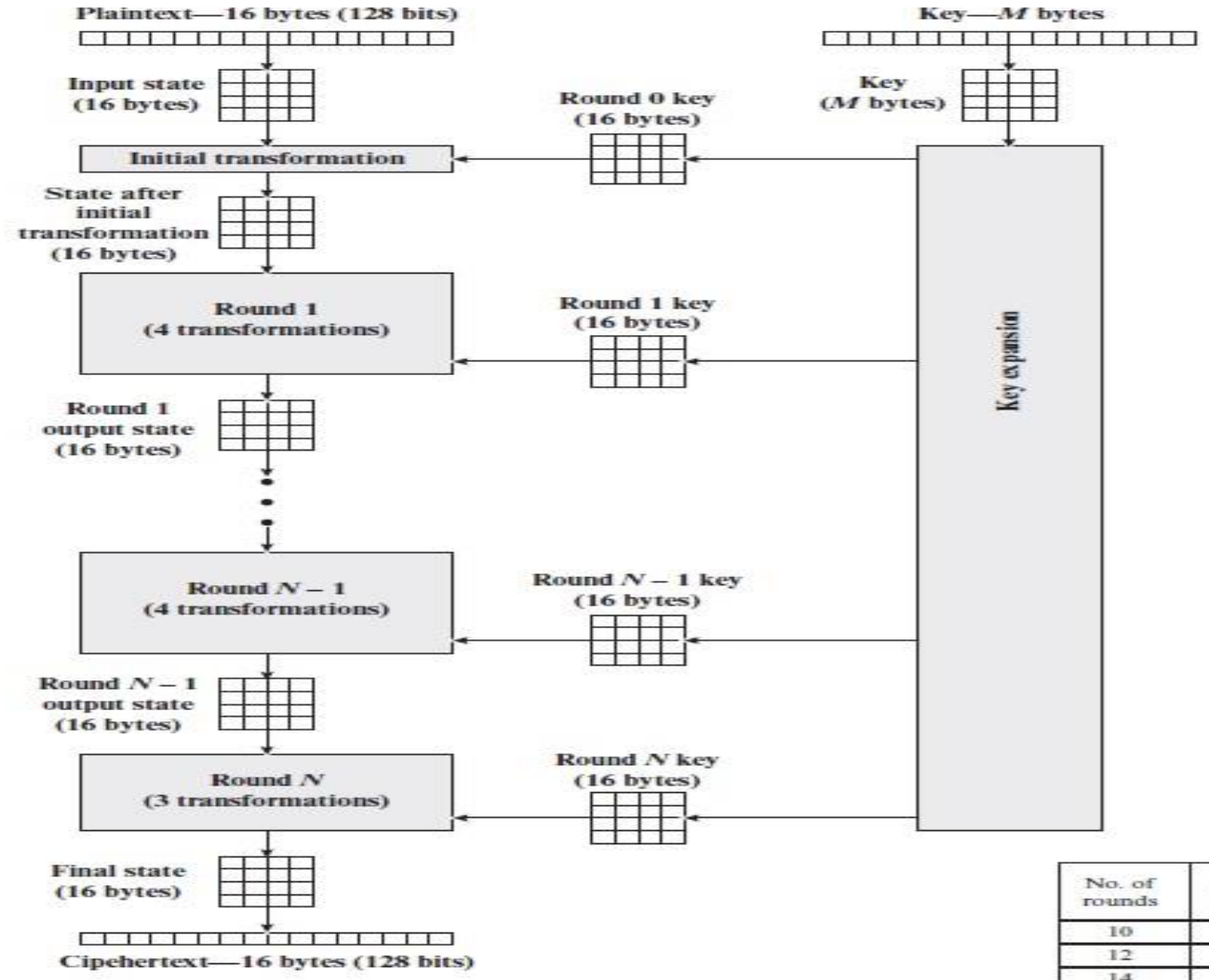
- The cipher key creates the first eight words (w_0 to w_7).
- If i mod 8 ≠ 0, $w_i \leftarrow w_{i-1} + w_{i-8}$; otherwise, $w_i \leftarrow t + w_{i-8}$.
- If i mod 4 = 0, but i mod 8 ≠ 0, then wi = SubWord $(w_{i-1}) + w_{i-8}$

Key-Expansion Analysis

The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

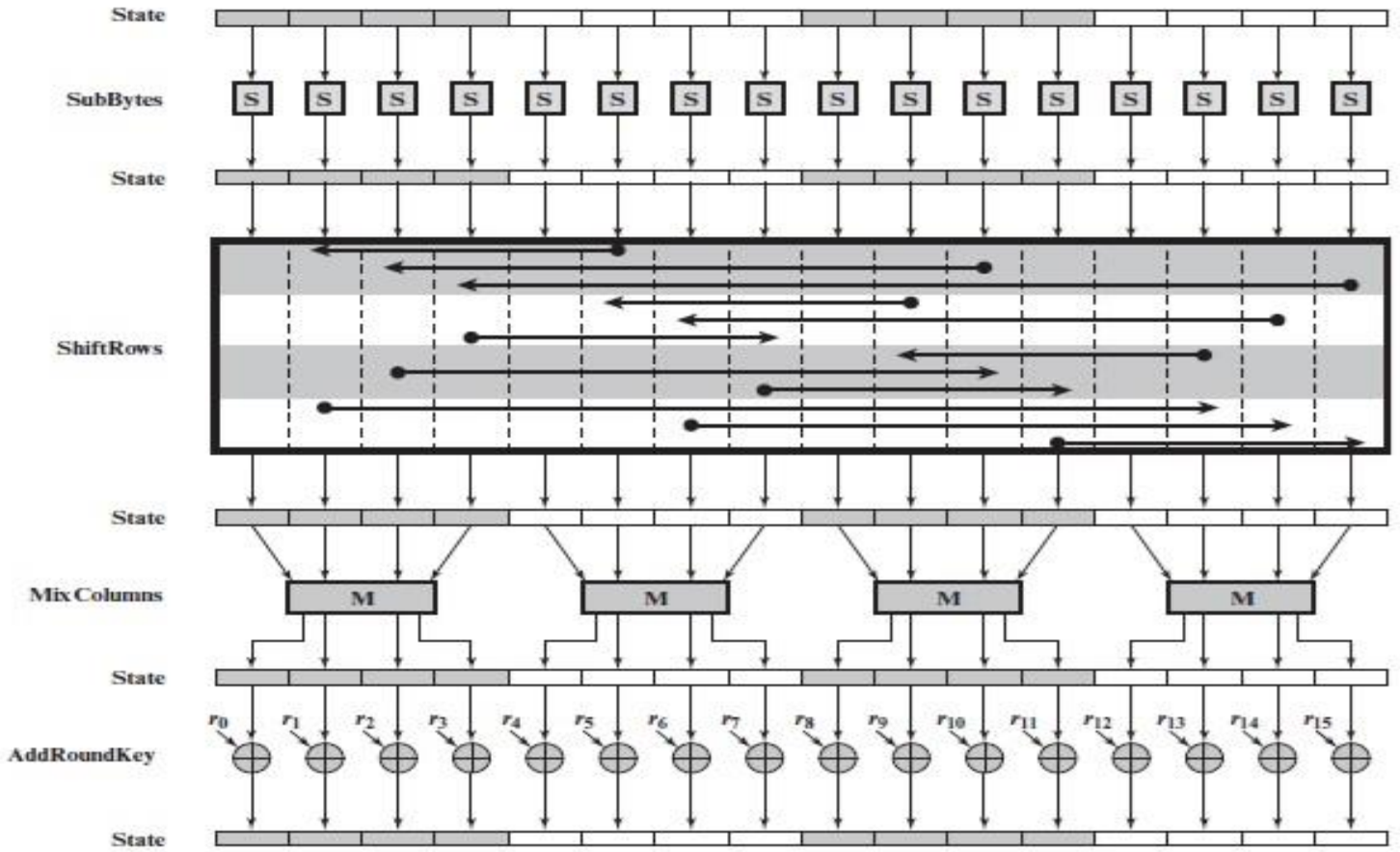
1. Even if Eve knows only part of the cipher key or the values of the words in some round keys, she still needs to find the rest of the cipher key before she can find all round keys. This is because of the nonlinearity produced by SubWord transformation in the key-expansion process.
2. Two different cipher keys, no matter how similar to each other, produce two expansions that differ in at least a few rounds.
3. Each bit of the cipher key is diffused into several rounds. For example, changing a single bit in the cipher key, will change some bits in several rounds.
4. The use of the constants, the RCons, removes any symmetry that may have been created by the other transformations.
5. There are no serious weak keys in AES, unlike in DES.
6. The key-expansion process can be easily implemented on all platforms.
7. The key-expansion routine can be implemented without storing a single table; all calculations can be done using the $GF(2^8)$ and FG(2) fields.

AES Encryption



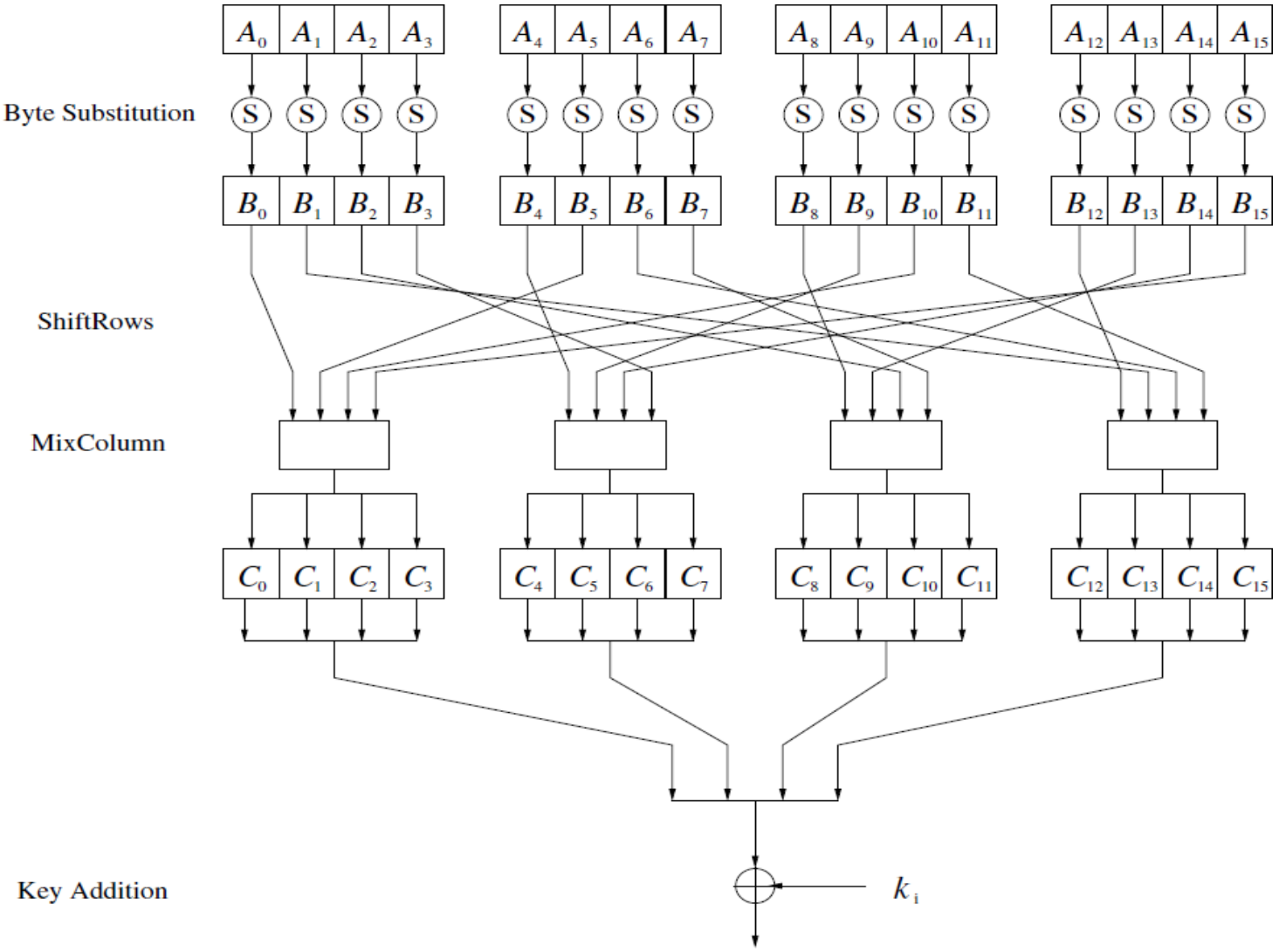
No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Encryption Round



AES Encryption Round

AES round function for rounds 1,2, . . . ,N_r–1



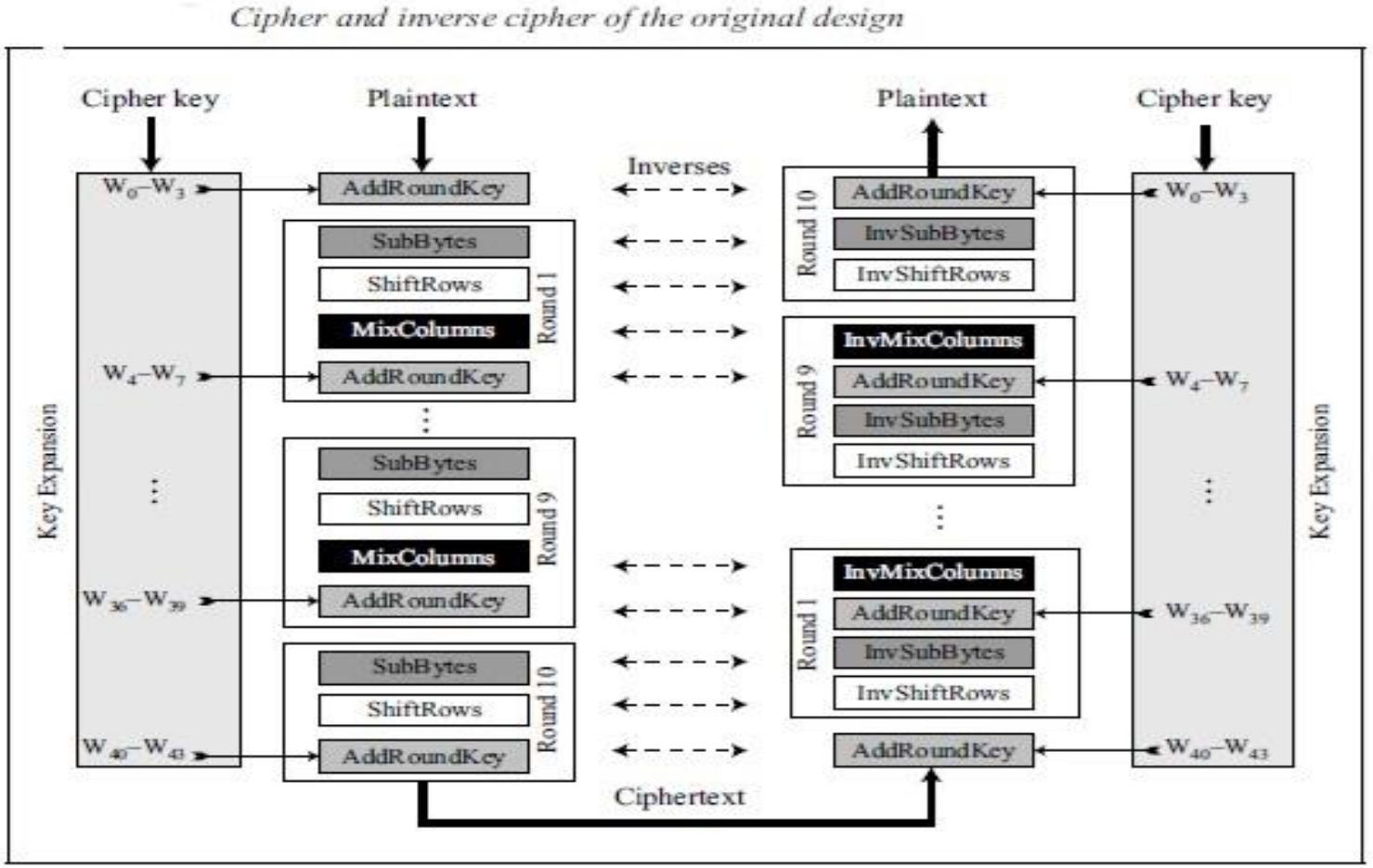
ShiftRows Sublayer

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

the output is the new state:

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	⟵ one position left shift
B_{10}	B_{14}	B_2	B_6	⟵ two positions left shift
B_{15}	B_3	B_7	B_{11}	⟵ three positions left shift

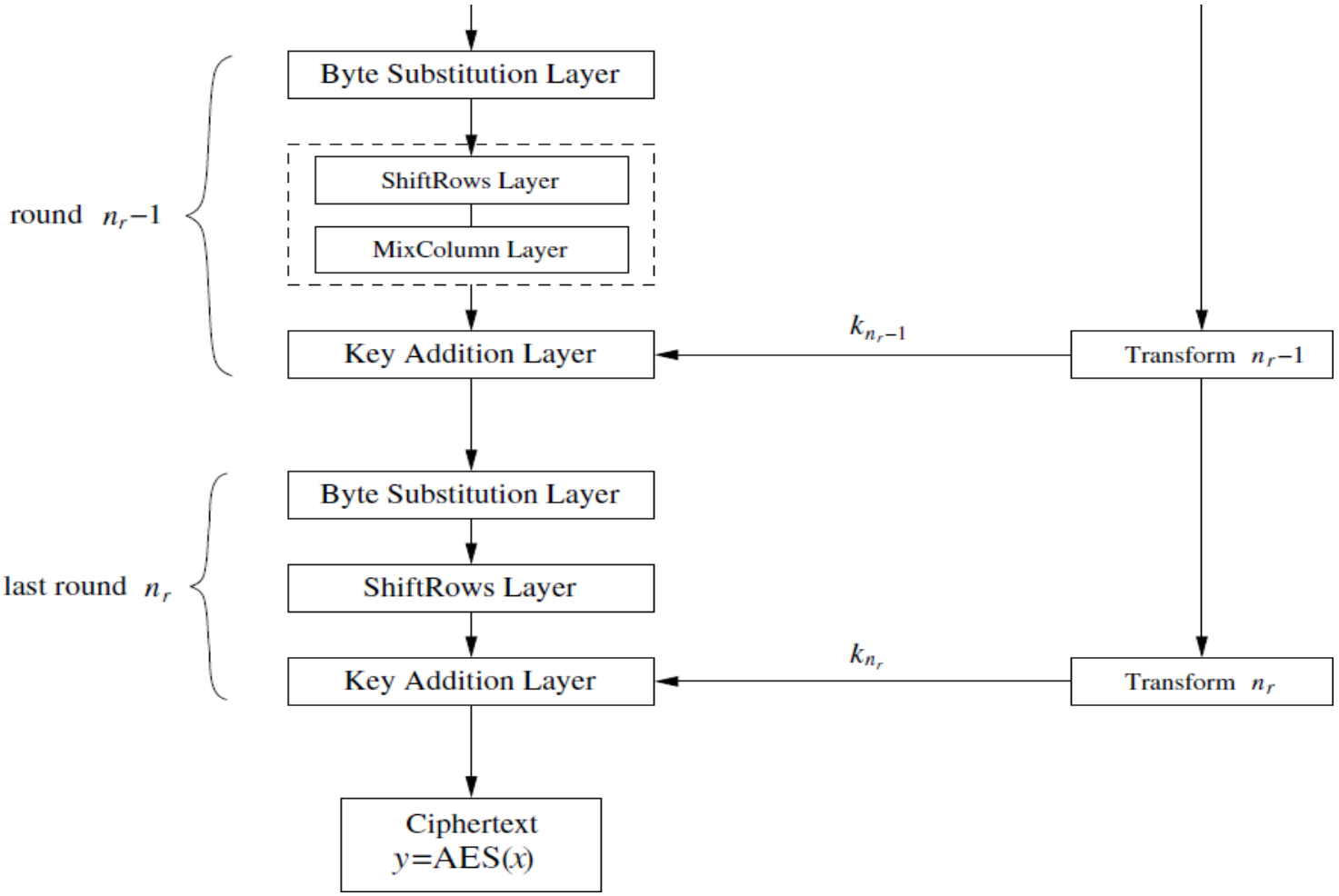
Original Design

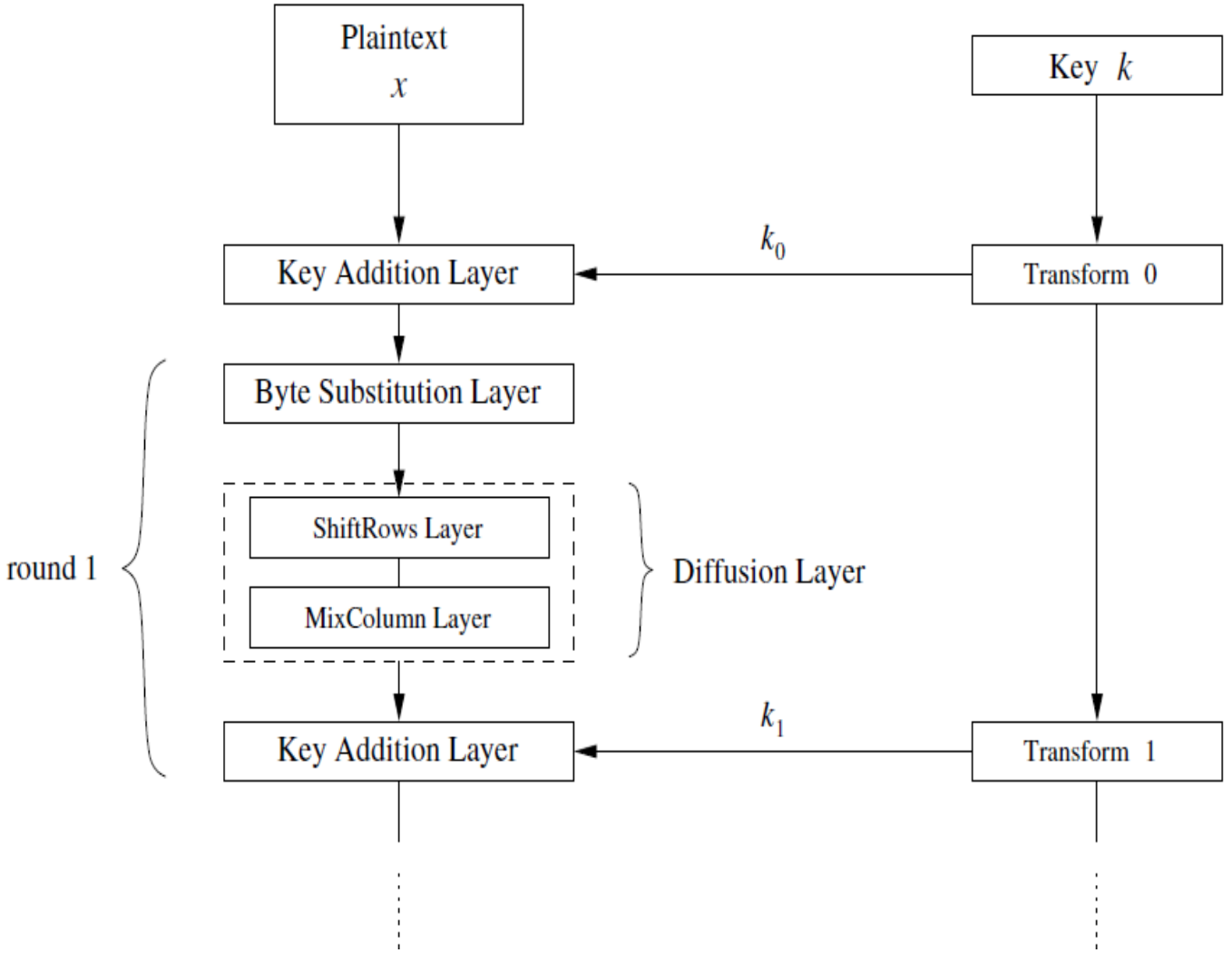


- The order of transformations in each round is not the same in the cipher and reverse cipher

- First, the order of SubBytes and ShiftRows is changed in the reverse cipher.
- Second, the order of MixColumns and AddRoundKey is changed in the reverse cipher.
- This difference in ordering is needed to make each transformation in the cipher aligned with its inverse in the reverse cipher.

- Consequently, the decryption algorithm as a whole is the inverse of the encryption algorithm.
- Round keys are used in the reverse order. Note that the encryption and decryption algorithms in the original design are not similar.


Byte Substitution Layer
ShiftRows Layer
MixColumn Layer
round n_r-1
Key Addition Layer
k_{n_r-1}
Transform n_r-1
Byte Substitution Layer
ShiftRows Layer
last round n_r
Key Addition Layer
k_{n_r}
Transform n_r
Ciphertext
y=AES(x)



AES is a non-Feistel cipher, which means that each transformation or group of transformations must be invertible

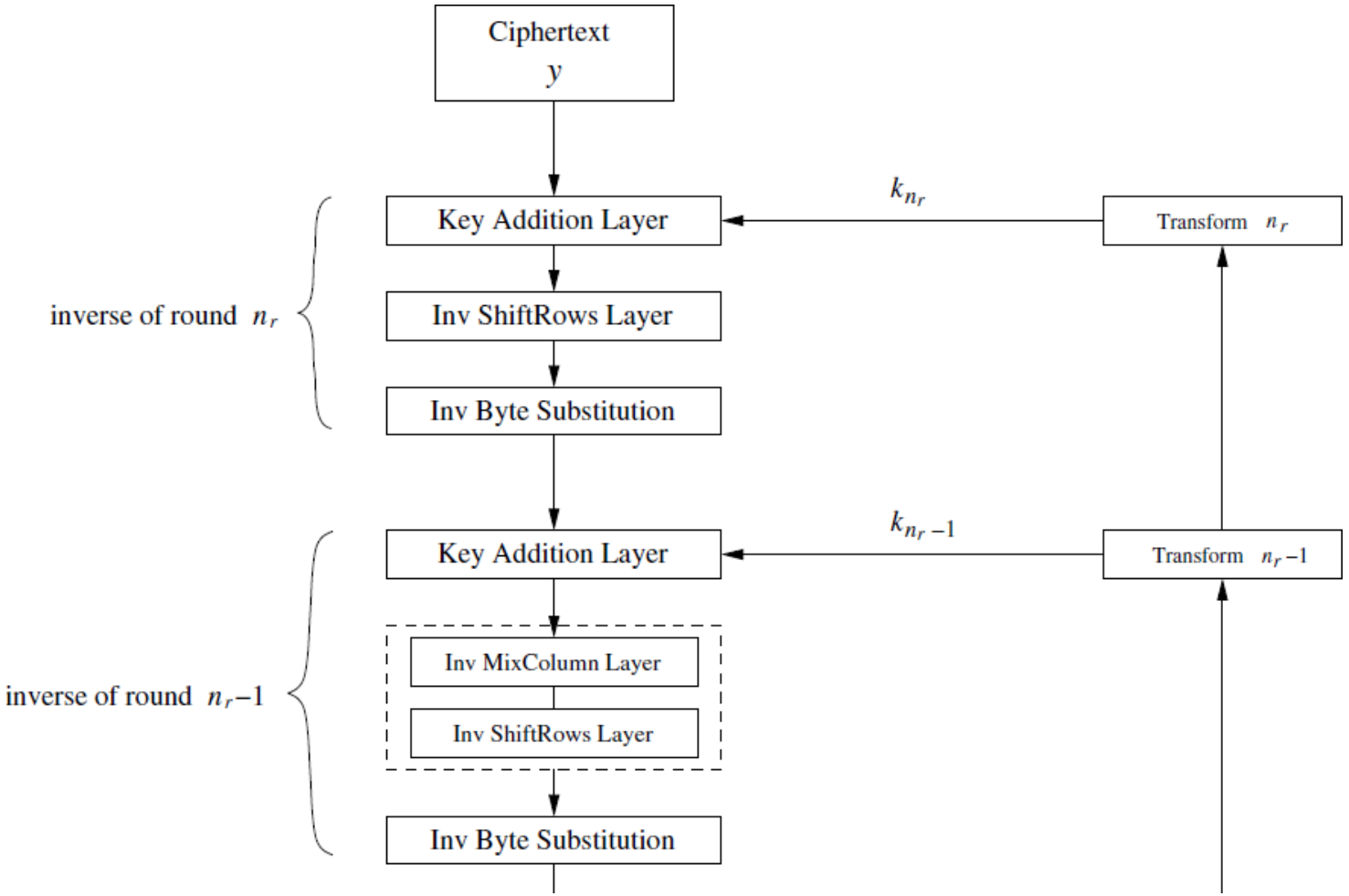
Pseudocode for cipher in the original design

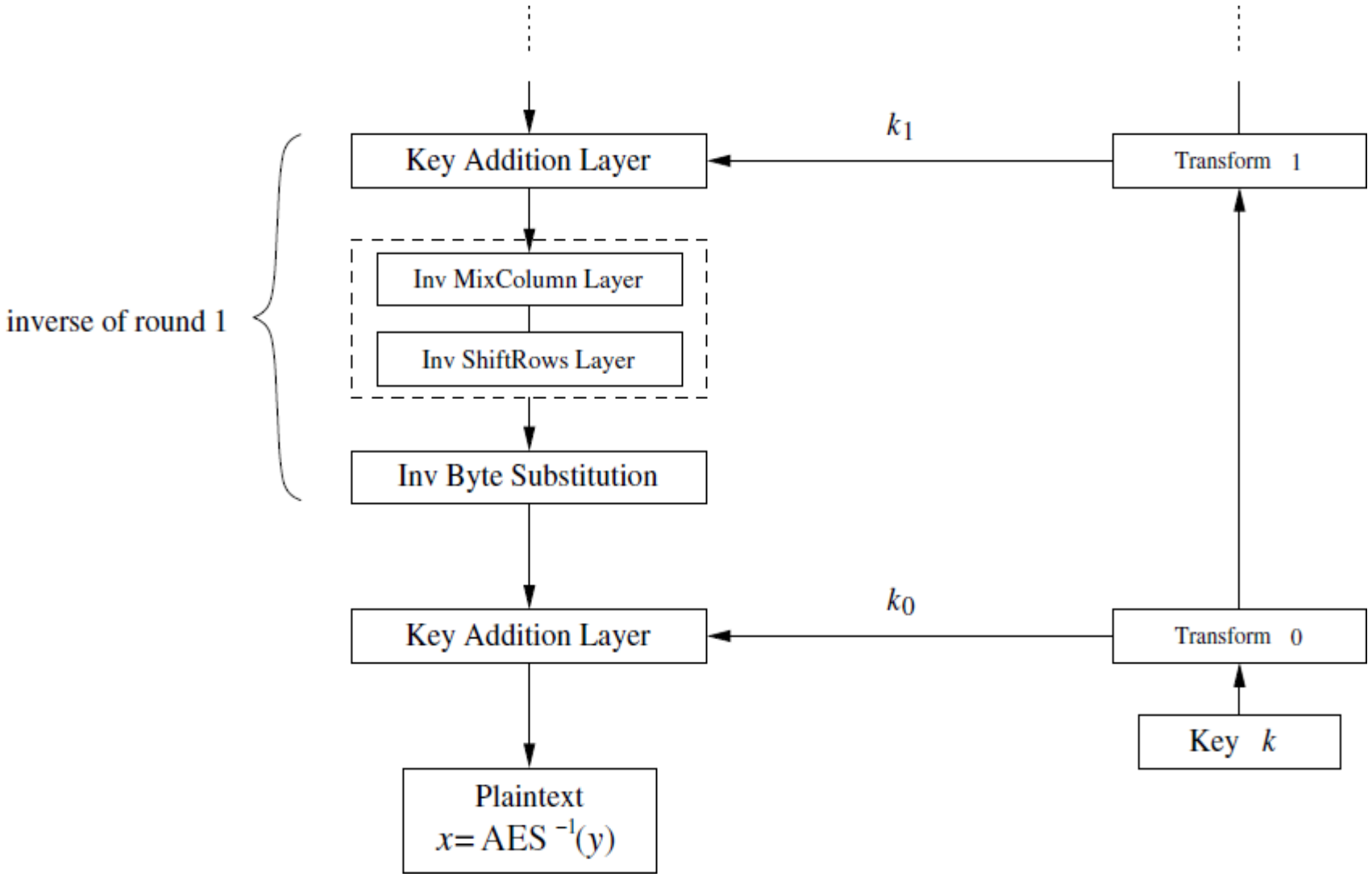
```
Cipher (InBlock [16], OutBlock[16], w[0 … 43])
{
   BlockToState (InBlock, S)

   S ← AddRoundKey (S, w[0…3])
    for (round = 1 to 10)
    {
        S ← SubBytes (S)
        S ← ShiftRows (S)
         if (round ≠ 10)  S ← MixColumns (S)
         S ← AddRoundKey (S, w[4 × round, 4 × round + 3])
    }

  StateToBlock (S, OutBlock);
}
```

AES Decryption

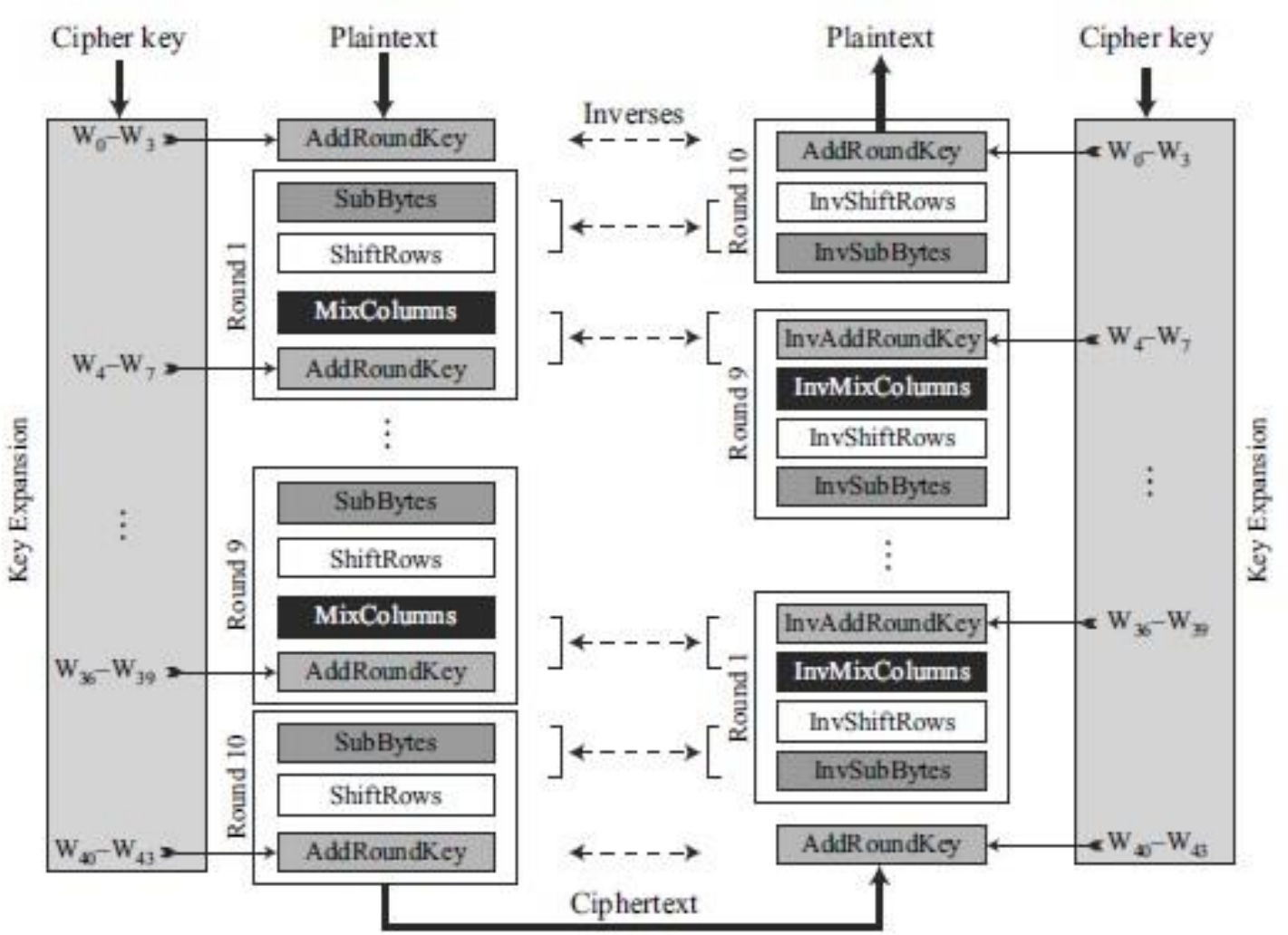
- AES is not based on a Feistel network.
- The Byte Substitution layer becomes the Inv Byte Substitution layer.
- The ShiftRows layer becomes the Inv ShiftRows layer, and the MixColumn layer becomes Inv MixColumn layer.
- The order of the subkeys is reversed.

AES decryption block diagram




Key Addition Layer
k_1
Transform 1
inverse of round 1
Inv MixColumn Layer
Inv ShiftRows Layer
Inv Byte Substitution
Key Addition Layer
k_0
Transform 0
Key k
Plaintext
$x = \mathrm{AES}^{-1}(y)$

Alternate


Cipher key
Plaintext
Plaintext
Cipher key
Inverses
Key Expansion
W_0–W_3
AddRoundKey
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round 1
W_4–W_7
SubBytes
ShiftRows
MixColumns
AddRoundKey
Round 9
W_{36}–W_{39}
SubBytes
ShiftRows
AddRoundKey
Round 10
W_{40}–W_{43}
Ciphertext
AddRoundKey
InvShiftRows
InvSubBytes
Round 10
InvAddRoundKey
InvMixColumns
InvShiftRows
InvSubBytes
Round 9
InvAddRoundKey
InvMixColumns
InvShiftRows
InvSubBytes
Round 1
AddRoundKey
W_0–W_3
W_4–W_7
W_{36}–W_{39}
W_{40}–W_{43}
Key Expansion

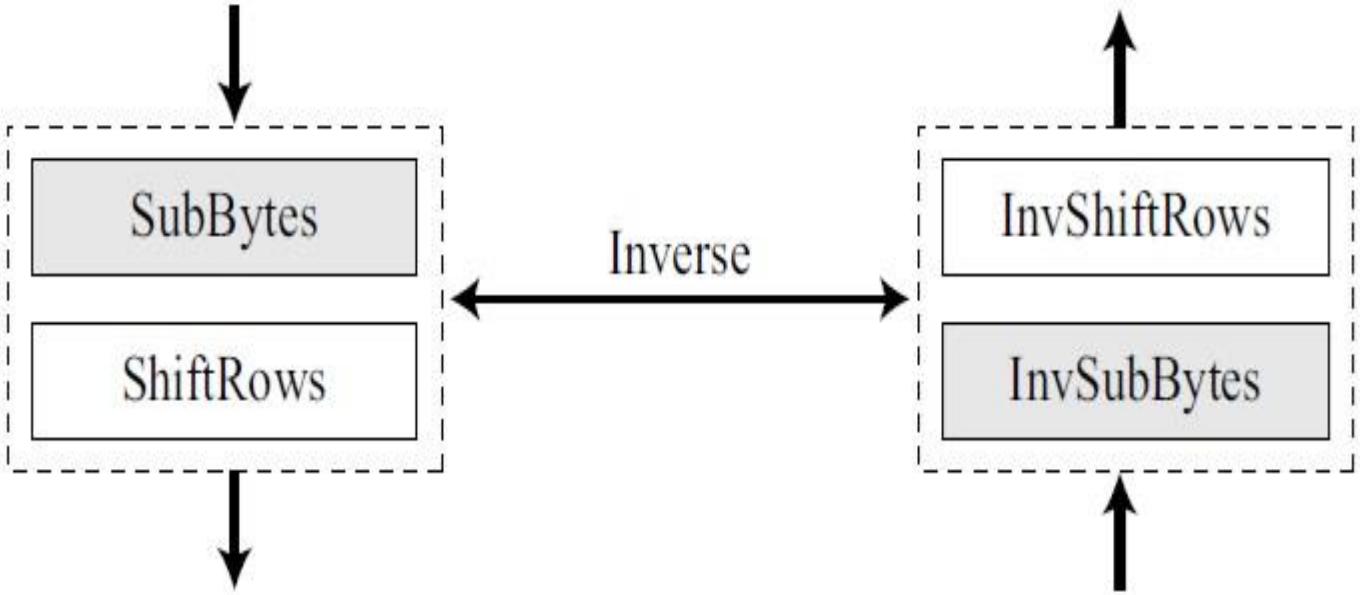
❑ SubBytes/ShiftRows Pairs

SubBytes change the contents of each byte without changing the order of the bytes in the state;
ShiftRows change the order of the bytes in the state without changing the contents of the bytes.

- This implies that we can change the order of these two transformations in the inverse cipher without affecting the invertibility of the whole algorithm.

- The combination of two transformations in the cipher and inverse cipher are the inverses of each other.

Invertibility of SubBytes and ShiftRows combinations

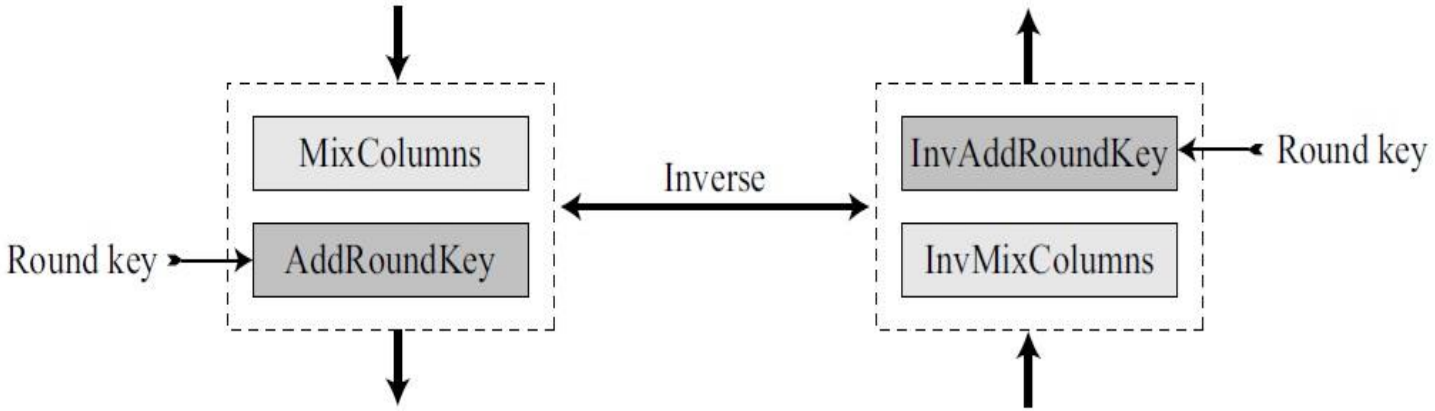


MixColumns/AddRoundKey Pair

Here the two involved transformations are of different nature.

However, the pairs can become inverses of each other if we multiply the key matrix by the inverse of the constant matrix used in MixColumns transformation.

We call the new transformation InvAddRoundKey.



It can be proved that the two combinations are now inverses of each other. In the cipher we call the input state to the combination S and the output state T. In the reverse cipher the input state to the combination is T. The following shows that the output state is also S. Note that the MixColumns transformation is actually multiplication of the C matrix (constant matrix by the state).

Cipher: $T = CS \oplus K$

Inverse Cipher: $C^{-1}T \oplus C^{-1}K = C^{-1}(CS \oplus K) \oplus C^{-1}K = C^{-1}CS \oplus C^{-1}K \oplus C^{-1}K = S$

- **Changing Key-Expansion Algorithm**

Instead of using InvRoundKey transformation in the reverse cipher, the key-expansion algorithm can be changed to create a different set of round keys for the inverse cipher.
However, note that the round key for the pre-round operation and the last round should not be changed. The round keys for rounds 1 to 9 need to be multiplied by the constant
matrix

Examples

The following shows the ciphertext block created from a plaintext block using a randomly selected cipher key.

Plaintext:	00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19
Cipher Key:	24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87
Ciphertext:	BC 02 8B D3 E0 E3 B1 95 55 0D 6D FB E6 F1 82 41

Analysis of AES

- **Security**

AES was designed after DES. Most of the known attacks on DES were already tested on AES; none of them has broken the security of AES so far.

- Brute-Force Attack

AES is definitely more secure than DES due to the larger-size key (128, 192, and 256 bits). Let us compare DES with 56-bit cipher key and AES with 128-bit cipher key. For DES we need 2^{56}(ignoring the key complement issue) tests to find the key; for AES we need 2^{128} tests to find the key. This means that if we can break DES in *t* seconds, we need ($2^{72} \times t$) seconds to break AES. This would be almost impossible. In addition, AES provides two other versions with longer cipher keys. The lack of weak keys is another advantage of AES over DES.

- Statistical Attacks

The strong diffusion and confusion provided by the combination of the SubBytes, ShiftRows, and MixColumns transformations removes any frequency pattern in the plaintext. Numerous tests have failed to do statistical analysis of the ciphertext.

- Differential and Linear Attacks

AES was designed after DES. Differential and linear cryptanalysis attacks were no doubt taken into consideration. There are no differential and linear attacks on AES as yet.

- AES was designed after DES
- Most of the known attacks on DES were already tested on AES.
- Brute-Force attack
- There are no differential and linear attacks on AES yet.
- It can be easily implemented using cheap processors.

Thank You