The Shift Rows Step (1)

- This is where the matrix representation of the state array becomes important.
- The ShiftRows transformation consists of (i) not shifting the first row of the state array at all; (ii) circularly shifting the second row by one byte to the left; (iii) circularly shifting the third row by two bytes to the left; and (iv) circularly shifting the last row by three bytes to the left.
- This operation on the state array can be represented by

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \Longrightarrow \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{bmatrix}$$

The Shift Rows Step (2)

- Recall again that the input block is written column-wise. That is the first four bytes of the input block fill the first column of the state array, the next four bytes the second column, etc. As a result, shifting the rows in the manner indicated scrambles up the byte order of the input block.
- For decryption, the corresponding step shifts the rows in exactly the opposite fashion. The first row is left unchanged, the second row is shifted to the right by one byte, the third row to the right by two bytes, and the last row to the right by three bytes, all shifts being circular.

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \Longrightarrow \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,3} & s_{1,0} & s_{1,1} & s_{1,2} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,1} & s_{3,2} & s_{3,3} & s_{3,0} \end{bmatrix}$$

The Mix Columns Step (1)

- This step replaces each byte of a column by a function of all the bytes in the same column.
- More precisely, each byte in a column is replaced by two times the value of that byte, plus three times the next byte, plus the byte that comes next, plus the byte that comes next. The word 'next' means the byte in the row below; the meaning of 'next' is circular in the same column.
- For the bytes in the first row of the state array, this operation can be stated as

$$s'_{0,j} = (2 \times s_{0,j}) \otimes (3 \times s_{1,j}) \otimes s_{2,j} \otimes s_{3,j}$$

- For the bytes in the second row of the state array, this operation can be stated as

$$s'_{1,j} = s_{0,j} \otimes (2 \times s_{1,j}) \otimes (3 \times s_{2,j}) \otimes s_{3,j}$$

The Mix Columns Step (2)

- For the bytes in the third row of the state array, this operation can be stated as

$$s'_{2,j} = s_{0,j} \otimes s_{1,j} \otimes (2 \times s_{2,j}) \otimes (3 \times s_{3,j})$$

- And, for the bytes in the fourth row of the state array, this operation can be stated as

$$s'_{3,j} = (3 \times s_{0,j}) \otimes s_{1,j} \otimes s_{2,j} \otimes (2 \times s_{3,j})$$

- More compactly, the column operations can be shown as

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 10 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_{0.0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1.0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2.0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3.0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0.0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1.0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2.0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3.0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

where, on the left hand side, when a row of the leftmost matrix multiples a column of the state array matrix, additions involved are meant to be XOR operations.

The Mix Columns Step (3)

- The corresponding transformation during decryption is given by

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

Adding the Round Key (1)

- The 128 bits of the state array are bitwise XOR'ed with the 128 bits of the round key.
- The AES Key Expansion algorithm is used to derive the 128-bit round key from the original 128-bit encryption key.
- In the same manner as the 128-bit input block is arranged in the form of a state array, the algorithm first arranges the 16 bytes of the encryption key in the form of a 4 × 4 array of bytes

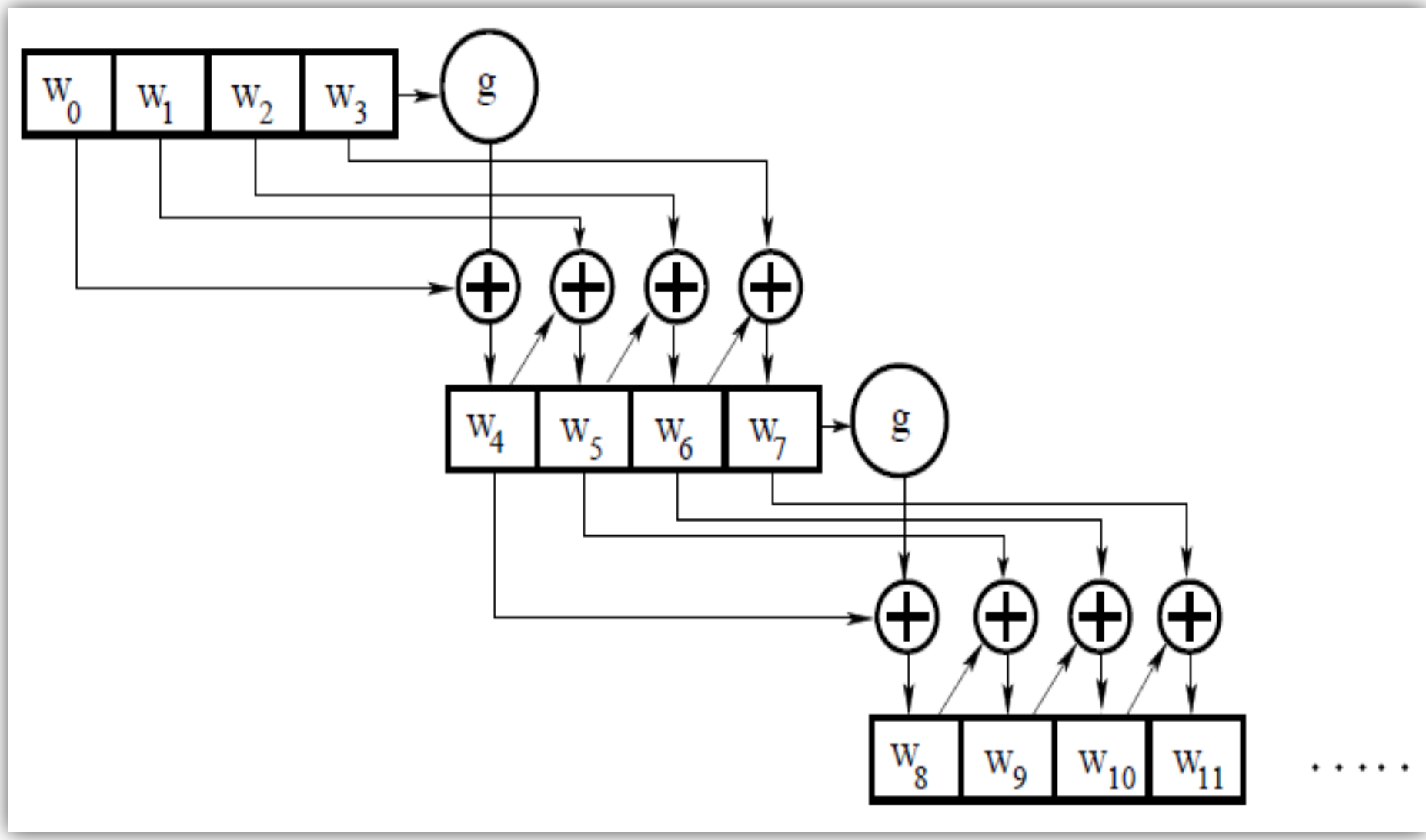
$$\begin{bmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{bmatrix} \Rightarrow \begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}$$

- The first four bytes of the encryption key constitute the word w_0, the next four bytes the word w_1, and so on.
- The algorithm subsequently expands the words [w_0, w_1, w_2, w_3] into a 44-word key schedule that can be labeled

  w_0, w_1, w_2, w_3, ..., w_{43}

Adding the Round Key (2)

- Of these, the words [w_0, w_1, w_2, w_3] are bitwise XOR'ed with the input block **before** the round-based processing begins.
- The remaining 40 words of the key schedule are used **four words at a time** in each of the 10 rounds.
- The above two statements are also true for decryption. The first four words of the key schedule are bitwise XOR'ed with the 128- bit ciphertext block before any round-based processing begins. Subsequently, each of the four words in the remaining 40 words of the key schedule are used in each of the ten rounds of processing.
- Now comes the difficult part: How does the **Key Expansion Algorithm** expand four words *w*0, *w*1, *w*2, *w*3 into the 44 words w_0, w_1, w_2, w_3, w_4, w_5, ..., w_{43} ?
- The key expansion will be explained with the help of the figure in the next slide. But first note that the key expansion takes place on a four-word to four-word basis, in the sense that each grouping of four words decides what the next grouping of four words will be.


w_0
w_1
w_2
w_3
g
w_4
w_5
w_6
w_7
g
w_8
w_9
w_{10}
w_{11}

The Algorithmic Steps in Going from a 4-Word Round Key to the Next 4-Word Round Key (1)

- Let's say that we have the four words of the round key:

  $w_i \quad w_{i+1} \quad w_{i+2} \quad w_{i+3}$

  Assuming that i is a multiple of 4, these will serve as the round key for the $(i/4)^{th}$ round. For example, w_4, w_5, w_6, w_7 is the round key for round 1, the sequence of words w_8, w_9, w_{10}, w_{11} the round key for round 2, and so on.
- Now we need to determine the words $w_{i+4} \quad w_{i+5} \quad w_{i+6} \quad w_{i+7}$ from the words $w_i \quad w_{i+1} \quad w_{i+2} \quad w_{i+3}$.
- From the previous figure, we write

  $$w_{i+5} = w_{i+4} \otimes w_{i+1}$$
  $$w_{i+6} = w_{i+5} \otimes w_{i+2}$$
  $$w_{i+7} = w_{i+6} \otimes w_{i+3}$$

  Note that except for the first word in a new 4-word grouping, each word is an XOR of the previous word and the corresponding word in the previous 4-word grouping.

The Algorithmic Steps in Going from a 4-Word Round Key to the Next 4-Word Round Key (2)

- So now we only need to figure out w_{i+4}. This is the beginning word of each 4-word grouping in the key expansion. The beginning word of each round key is obtained by:

$$w_{i+4} = w_i \otimes g(w_{i+3})$$

- That is, the first word of the new 4-word grouping is to be obtained by XOR'ing the first word of the last grouping with what is returned by applying a function *g* ( ) to the last word of the previous 4-word grouping.
- The function *g* ( ) consists of the following three steps:

1. Perform a one-byte left circular rotation on the argument 4-byte word.
2. Perform a byte substitution for each byte of the word returned by the previous step by using the same 16 × 16 lookup table as used in the SubBytes step of the round.
3. XOR the bytes obtained from the previous step with what is known as a round constant. The round constant is a word whose three rightmost bytes are always zero. Therefore, XOR'ing with the round constant amounts to XOR'ing with just its leftmost byte.

The Algorithmic Steps in Going from a 4-Word Round Key to the Next 4-Word Round Key (3)

- The round constant for the i^{th} round is denoted $Rcon[i]$. Since, by specification, the three rightmost bytes of the round constant are zero, we can write it as shown below. The left hand side of the equation below stands for the round constant to be used in the i^{th} round. The right hand side of the equation says that the rightmost three bytes of the round constant are zero.

$$Rcon[i] = (RC[i], 0, 0, 0)$$

- The only non-zero byte in the round constants, $RC[i]$, obeys the following recursion:

$$RC[1] = 1$$

$$RC[j] = 2 \times RC[j-1]$$

- The addition of the round constants is for the purpose of destroying any symmetries that may have been introduced by the other steps in the key expansion algorithm.

Final notes on the AES Key Expansion Algorithm

- In the AES Key Expansion Algorithm, if you change one bit of the encryption key, it will affect the round key for several rounds.
- The key expansion algorithm ensures that AES has no weak keys. A weak key is a key that reduces the security of a cipher in a predictable manner.
- For example, DES is known to have weak keys. Weak keys of DES are those that produce identical round keys for each of the 16 rounds. An example of DES weak key is when it consists of alternating ones and zeros. This sort of a weak key in DES causes all the round keys to become identical, which, in turn, causes the encryption to become self-inverting. That is, plain text encrypted and then encrypted again will lead back to the same plain text.
- (Since the small number of weak keys of DES are easily recognized, it is not considered to be a problem with that cipher.)

Finally . . .

- Acknowledgment: These lecture notes are based on the textbook by William Stallings and notes prepared by Avinash Kak, Purdue University. My sincere thanks are devoted to them and to all other people who offered the material on the web.

- Students are advised to study and solve the problems and answer the questions in **Assignment-A4**.