Advanced Encryption Standard (AES)

Tsay, Yih-Kuen

Dept. of Information Management

National Taiwan University

The Origin of AES

- A symmetric cipher intended to replace DES and 3DES (DES is slow and 3DES is three times as slow. Both use a 64-bit block size; a larger block size would be more secure.)
- A call for proposals for a new Advanced Encryption Standard issued in 1997 by NIST
- Selected algorithm: Rijndael, designed by Joan Daemen and Vincent Rijmen from Belgium.
- Published as FIPS PUB 197 in November, 2001
- Block size: 128 bits; key lengths: 128, 192, and 256 bits

Evaluation Criteria for AES

SECURITY

- **Actual security:** compared to other submitted algorithms (at the same key and block size).
- **Randomness:** the extent to which the algorithm output is indistinguishable from a random permutation on the input block.
- **Soundness:** of the mathematical basis for the algorithm's security.
- **Other security factors:** raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.

COST

- **Licensing requirements:** NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- **Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of the algorithm. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.
- **Memory requirements:** The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

Source: Table 5.1, Stallings 2006

Evaluation Criteria for AES (cont.)

ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

- **Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include (but are not limited to) the following:
  - **a.** The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.])
  - **b.** The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).
  - **c.** The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.
- **Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.
- **Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

Source: Table 5.1, Stallings 2006

Final Evaluation of Rijndael

General Security

Rijndael has no known security attacks. Rijndael uses S-boxes as nonlinear components. Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.

Software Implementations

Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance even when implemented in a mode not capable of interleaving. Rijndael's key setup time is fast.

Restricted-Space Environments

In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

Hardware Implementations

Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

Source: Table 5.2, Stallings 2006

Final Evaluation of Rijndael (cont.)

Attacks on Implementations

The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists, and its RAM requirement remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.

Encryption vs. Decryption

The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than for encryption.

Key Agility

Rijndael supports on-the-fly subkey computation for encryption. Rijndael requires a one-time execution of the key schedule to generate all subkeys prior to the first decryption with a specific key. This places a slight resource burden on the key agility of Rijndael.

Other Versatility and Flexibility

Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

Potential for Instruction-Level Parallelism

Rijndael has an excellent potential for parallelism for a single block encryption.

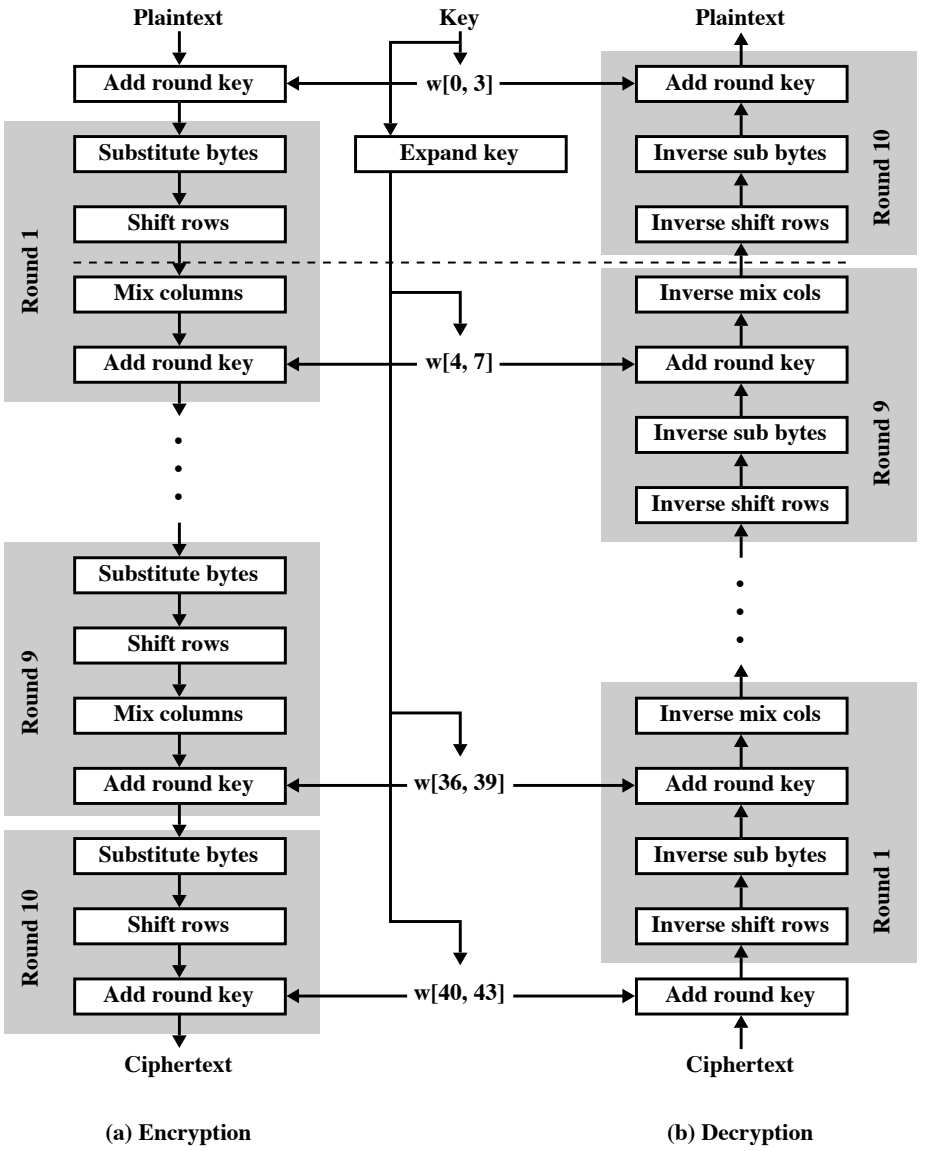
Source: Table 5.2, Stallings 2006

AES Parameters

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

Source: Table 5.3, Stallings 2006

AES Encryption and Decryption

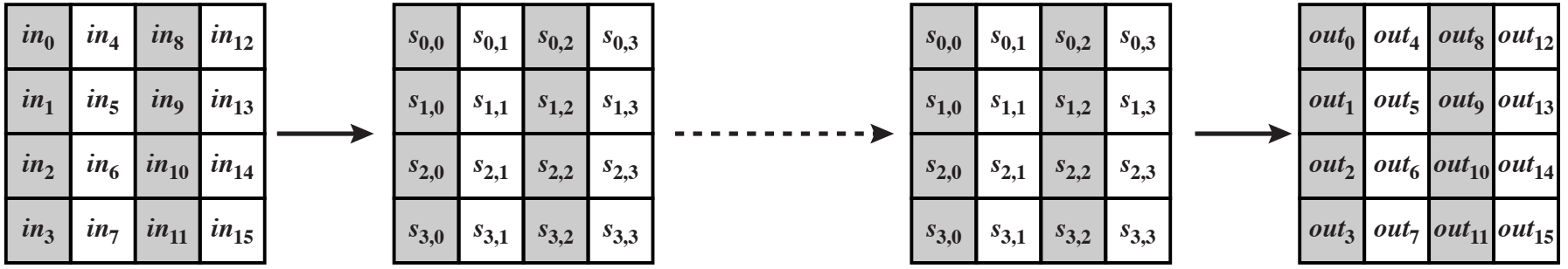


Source: Figure 5.1, Stallings 2006

About the AES Cipher

- Not a Feistel structure; entire data block processed in each round
- Input key expanded into 11 round keys of the same length
- Four stages used: **Substitute bytes**, **Shift rows** (the only permutation), **Mix columns**, **Add round key**
- Decryption algorithm different from encryption algorithm
- Correctness easy to verify.

AES Data Structures

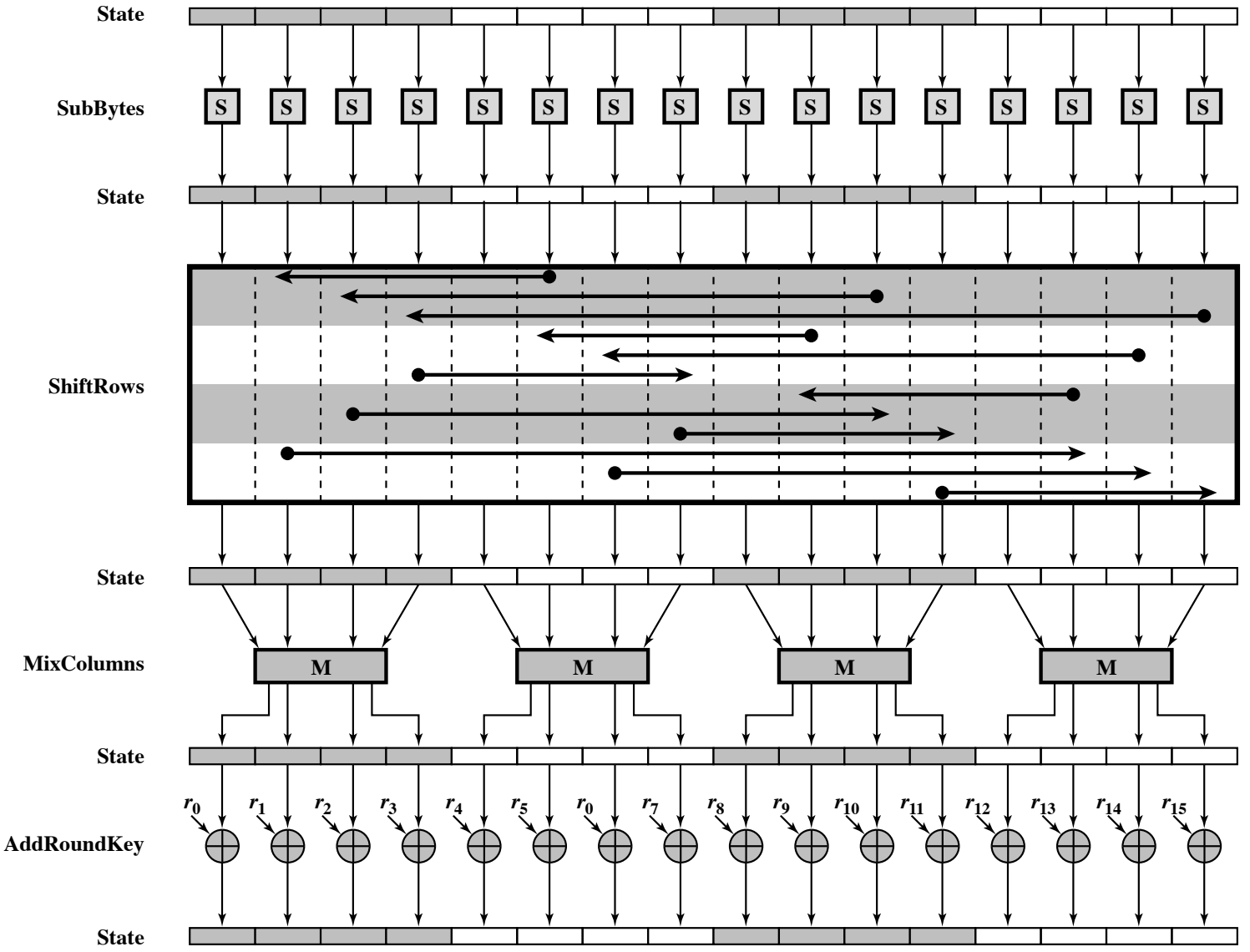


(a) Input, state array, and output

(b) Key and expanded key

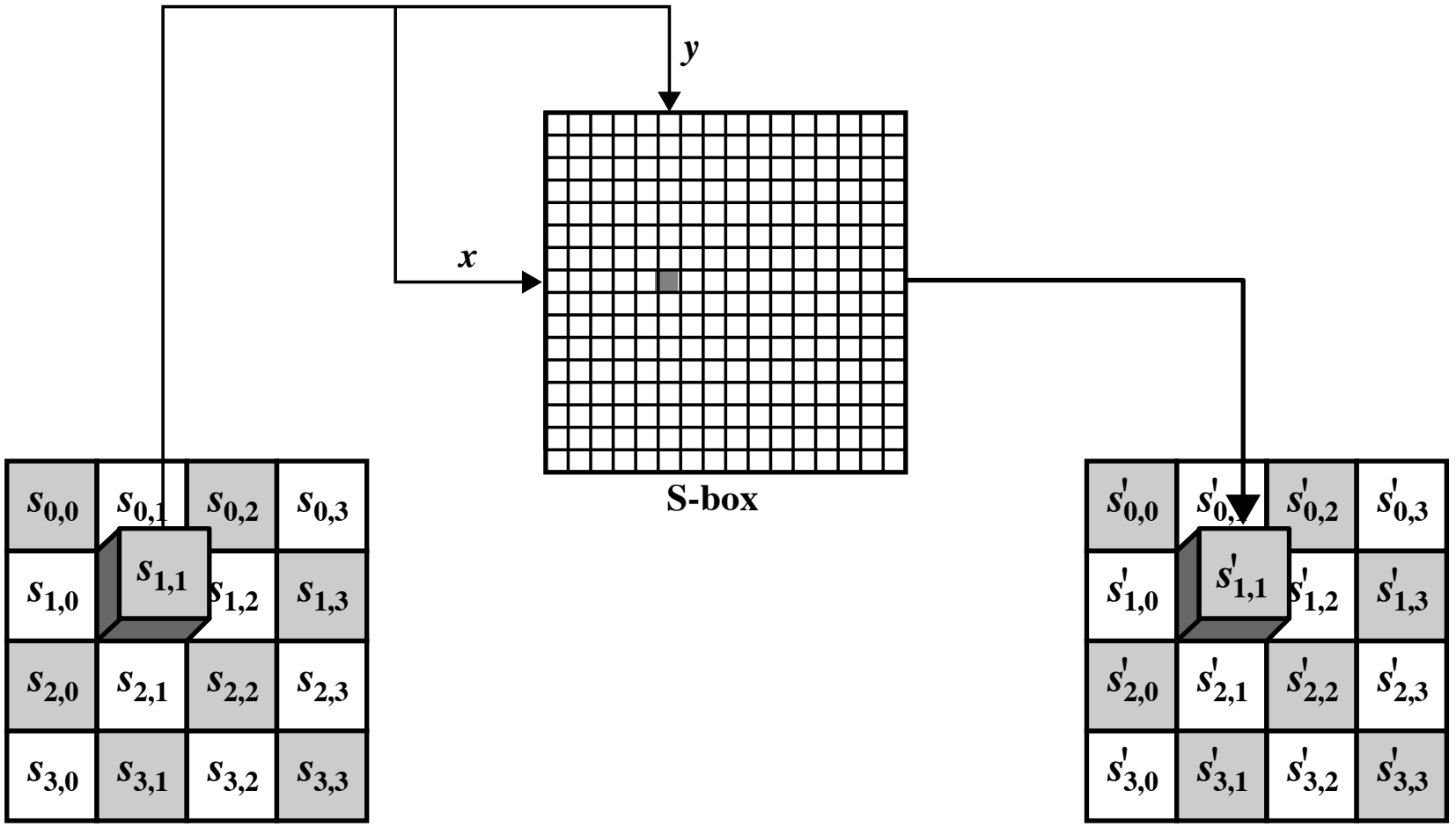
Source: Figure 5.2, Stallings 2006

AES Encryption Round



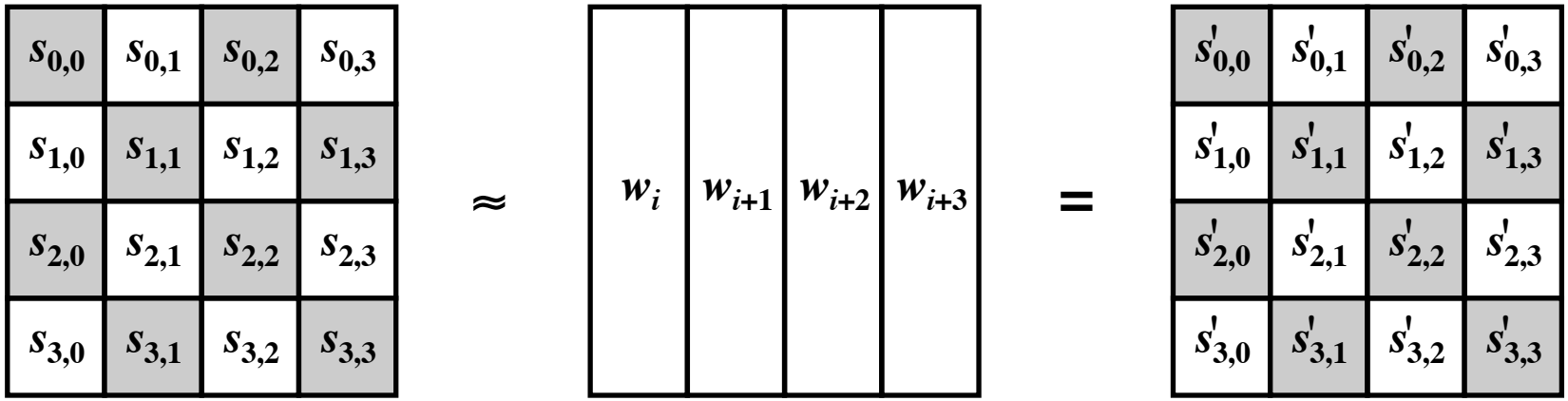
Source: Figure 5.3, Stallings 2006

AES Byte-Level Operations



Source: Figure 5.4(a), Stallings 2006

AES Byte-Level Operations (cont.)



Source: Figure 5.4(b), Stallings 2006

AES S-Boxes

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Source: Table 5.4, Stallings 2006

AES S-Boxes (cont.)

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Source: Table 5.4, Stallings 2006

An Example of SubBytes

EA	04	65	85
83	45	$5D$	96
$5C$	33	98	$B0$
$F0$	$2D$	AD	$C5$

$\rightarrow$

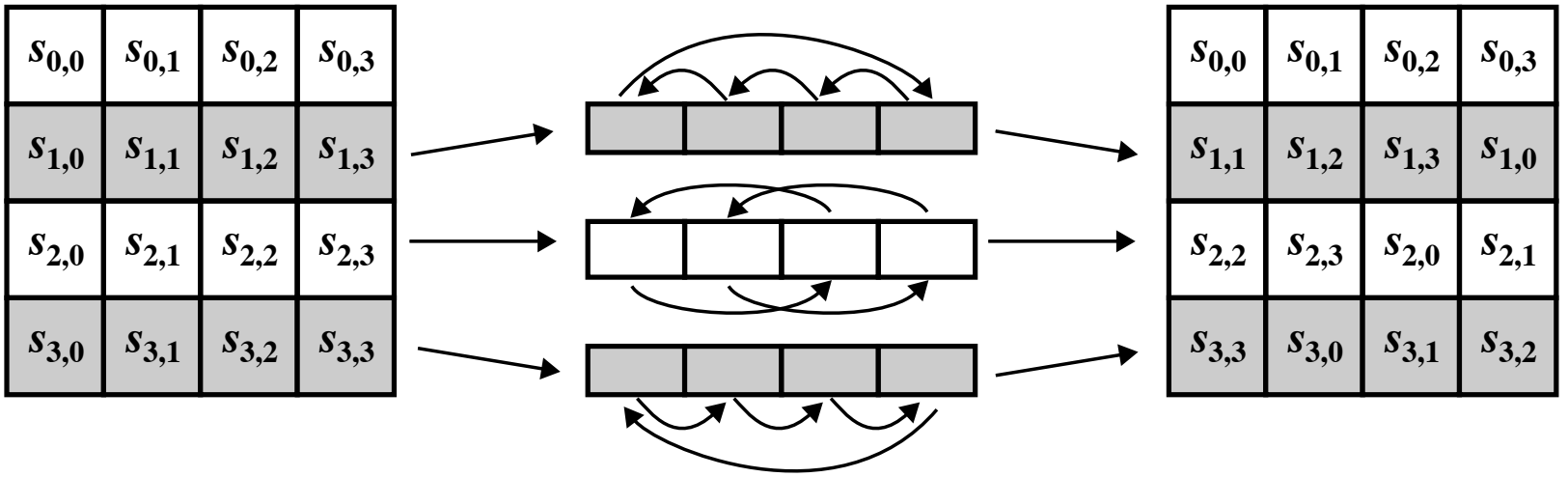
87	$F2$	$4D$	97
EC	$6E$	$4C$	90
$4A$	$C3$	46	$E7$
$8C$	$D8$	95	$A6$

Construction of the S-Box

- Initialization: 1st row: $\{00\}$, $\{01\}$, $\{02\}$, $\cdots$, $\{0F\}$; 2nd row: $\{10\}$, $\{11\}$, $\{12\}$, $\cdots$, $\{1F\}$; etc.
- Replace each byte with its multiplicative inverse; the value $\{00\}$ is mapped to itself.
- Apply the following (invertible) transformation:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Shift Rows



Source: Figure 5.5(a), Stallings 2006

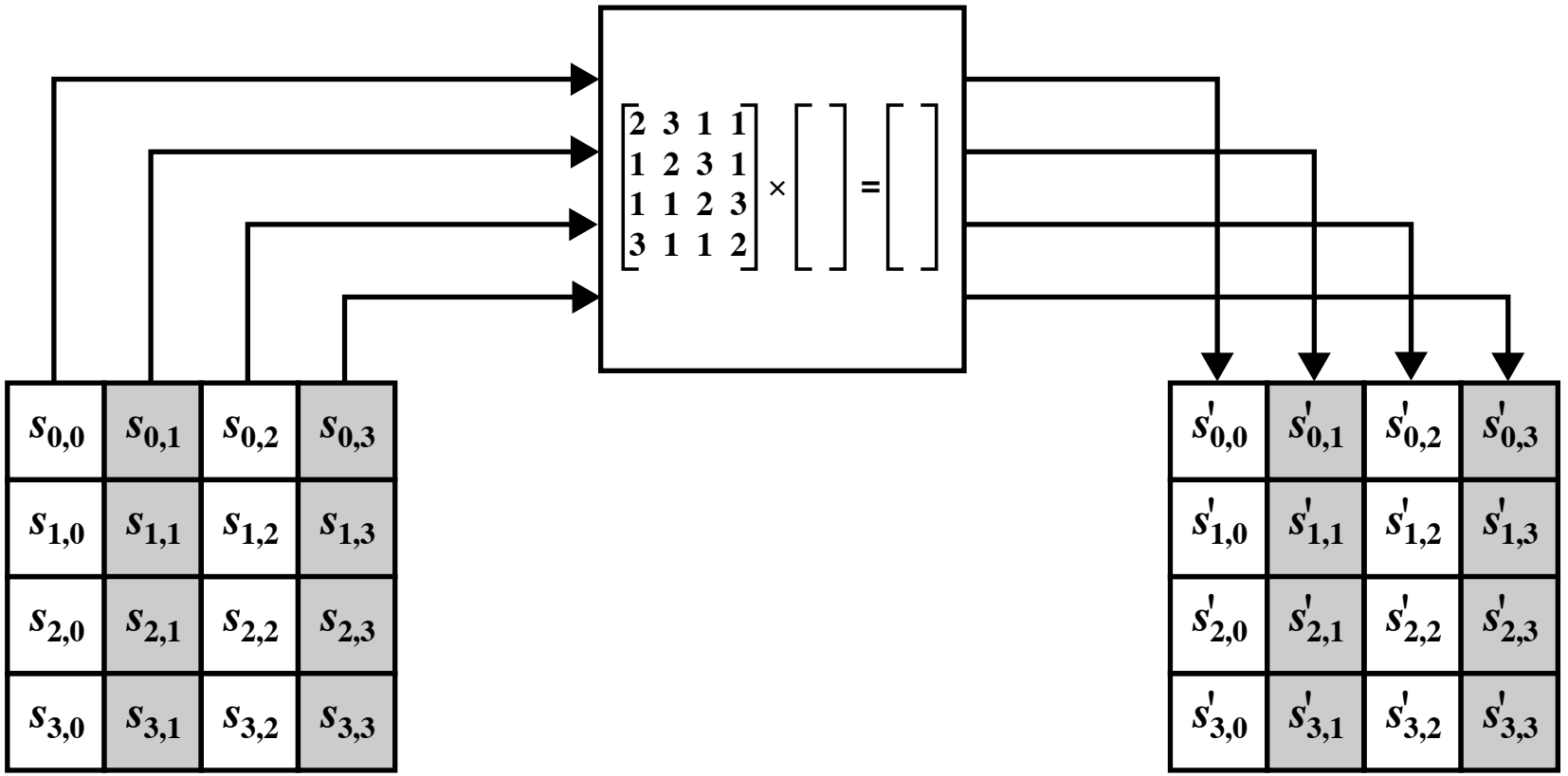
An Example of `ShiftRows`

87	$F2$	$4D$	97
EC	$6E$	$4C$	90
$4A$	$C3$	46	$E7$
$8C$	$D8$	95	$A6$

$\rightarrow$

87	$F2$	$4D$	97
$6E$	$4C$	90	EC
46	$E7$	$4A$	$C3$
$A6$	$8C$	$D8$	95

Mix Columns



Source: Figure 5.5(b), Stallings 2006

An Example of MixColumns

87	$F2$	$4D$	97
$6E$	$4C$	90	EC
46	$E7$	$4A$	$C3$
$A6$	$8C$	$D8$	95

$\rightarrow$

47	40	$A3$	$4C$
37	$D4$	70	$9F$
94	$E4$	$3A$	42
ED	$A5$	$A6$	BC

$$
\begin{aligned}
(\{02\} \bullet \{87\}) &= 00010101 \\
(\{03\} \bullet \{6E\}) &= 10110010 \\
\{46\} &= 01000110 \\
\{A6\} &= 10100110 \\
\hline
&\quad 01000111\ (= \{47\})
\end{aligned}
$$

InvMixColumns

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

An Example of AddRoundKey

$$\begin{array}{|c|c|c|c|} \hline 47 & 40 & A3 & 4C \\ \hline 37 & D4 & 70 & 9F \\ \hline 94 & E4 & 3A & 42 \\ \hline ED & A5 & A6 & BC \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline AC & 19 & 28 & 57 \\ \hline 77 & FA & D1 & 5C \\ \hline 66 & DC & 29 & 00 \\ \hline F3 & 21 & 41 & 6A \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|} \hline EB & 59 & 8B & 1B \\ \hline 40 & 2E & A1 & C3 \\ \hline F2 & 38 & 13 & 42 \\ \hline 1E & 84 & E7 & D2 \\ \hline \end{array}$$

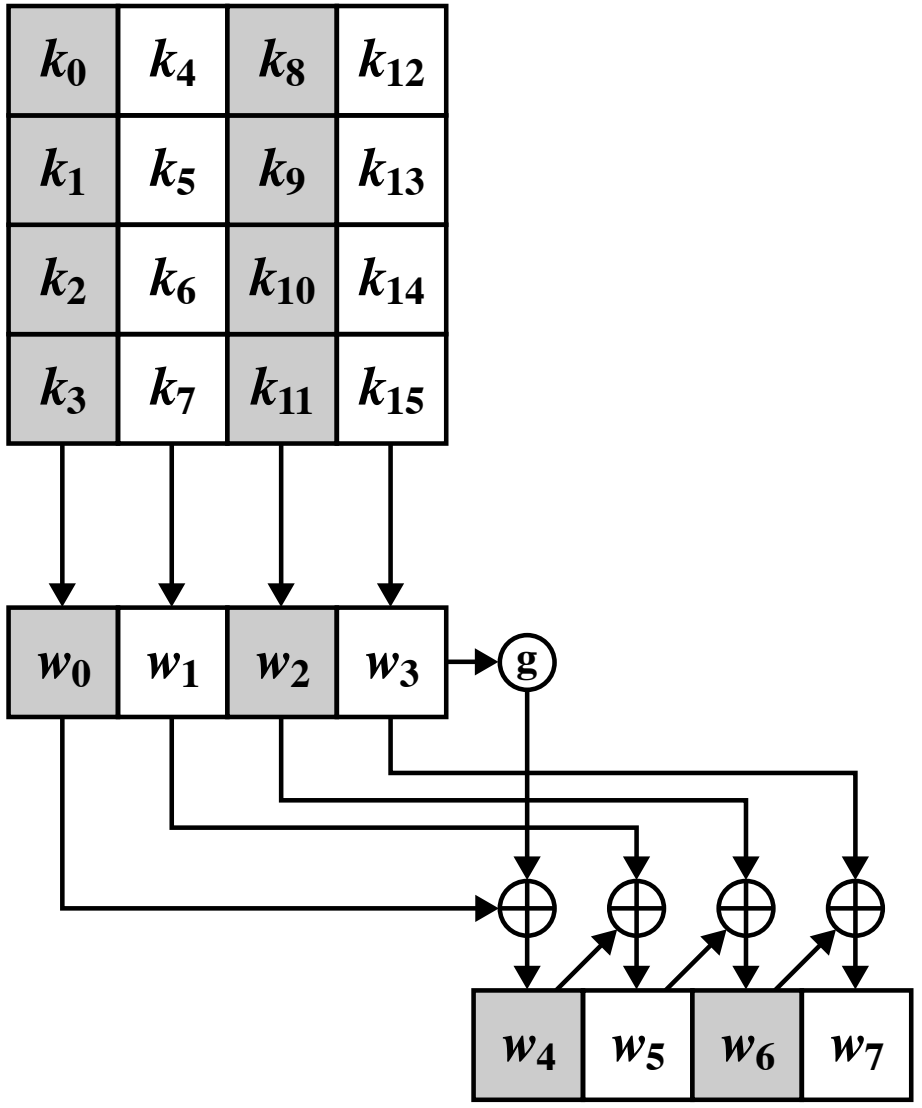
Key Expansion

KeyExpansion (**byte** key[16],**word** w[44])
{
word temp
for ($i = 0$; $i < 4$; $i++$)
w[i] = (key[$4 * i$],key[$4 * i + 1$],key[$4 * i + 2$],key[$4 * i + 3$]);
for ($i = 4$; $i < 44$; $i++$)
{ temp = w[$i - 1$];
if ($i \bmod 4 = 0$) temp = SubWord(RotWord(temp))$\oplus$Rcon[$i/4$];
w[i] = w[$i - 4$]$\oplus$temp
}
}

Rcon[j] = (RC[j],0,0,0), with RC[1]=1, RC[j]=2•RC[$j - 1$]

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

AES Key Expansion



Source: Figure 5.6, Stallings 2006

An Example of Key Expansion

Suppose the round key (Words 32, 33, 34, and 35) for Round 8 is

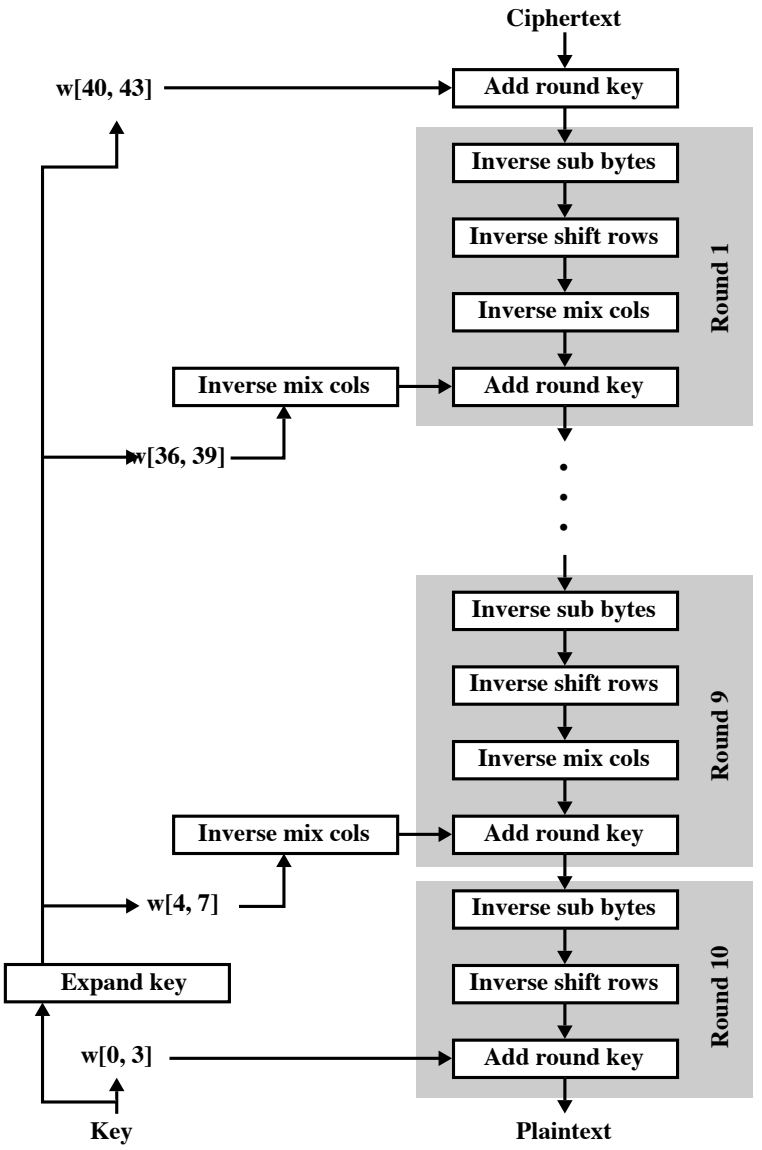
EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F.

The first 4 bytes (Word 36) of the round key for round 9 are calculated as follows:

i	temp	RotWord	SubWord	Rcon(9)
36	7F8D292F	8D292F7F	5DA515D2	1B000000

XOR	w[$i-4$]	w[i]
46A515D2	EAD27321	AC7766F3

Equivalent Inverse Cipher



Source: Figure 5.7, Stallings 2006

Equivalent Inverse Cipher (cont.)

- Interchanging **InvShiftRows** and **InvSubBytes**:

  InvShiftRows[InvSubBytes(S_i)] =
  InvSubBytes[InvShiftRows(S_i)]

- Interchanging **AddRoundKey** and **InvMixColumns**:
  For a given state S_i and a given round key w_j,

  InvMixColumns($S_i \oplus w_j$)
  = [InvMixColumns(S_i)] $\oplus$ [InvMixColumns(w_j)]

Implementation in 32-Bit Processes

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j+1}] \\ S[a_{2,j+2}] \\ S[a_{3,j+3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} =$$

$$\left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[a_{1,j+1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[a_{2,j+2}] \right) \oplus$$

$$\left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[a_{3,j+3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Implementation in 32-Bit Processes (cont.)

To facilitate the preceding calculation, four tables may be defined:

$$T_0(x) = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[x] \right); \quad T_1(x) = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[x] \right)$$

$$T_2(x) = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[x] \right); \quad T_3(x) = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[x] \right)$$

These tables can be pre-computed.