ADVANCED ENCRYPTION STANDARD
The Origins of AES

The principal drawback of 3DES (which was recommended in 1999, Federal Information Processing Standard FIPS PUB 46-3 as new standard with 168-bit key) is that the algorithm is relatively sluggish in software. A secondary drawback is the use of 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

In 1997, National Institute of Standards and Technology NIST issued a call for proposals for a new Advanced Encryption Standard (AES), which should have security strength equal to or better than 3DES, and significantly improved efficiency. In addition, NIST also specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits.

In a first round of evaluation, 15 proposed algorithms were accepted. A 2nd round narrowed to 5 algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November, 2001. NIST selected Rijndael as the proposed AES algorithm. The 2 researches of AES are Dr. Joan Daemon and Dr. Vincent Rijmen from Belgium.

AES Evaluation

Security – 128 minimal key size provides enough security

Cost – AES should have high computational efficiency

The Origins of AES (Cont 1)

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 1 of 2)

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.

Algorithm and implementation characteristics – this includes variety of considerations, including flexibility, suitability for hardware and software implementations, simplicity

The Origins of AES (Cont 2)

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 2 of 2)

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.

Additional criteria include: general security, software implementations, restricted-space environments, hardware implementations, attacks on implementation (timing attacks), encryption versus decryption, key agility, flexibility, potential for instruction-level parallelism.

The Origins of AES (Cont 3)

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 1 of 2)

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.

The Origins of AES (Cont 4)

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 2 of 2)

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.

THE AES CIPHER

Table 5.3 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

A number of AES parameters depend on the key length (Table 5.3). In the description of this section, we assume the key length of 128 bits.
Figure 5.1 shows the overall structure of AES.

OVERALL STRUCTURE OF AES

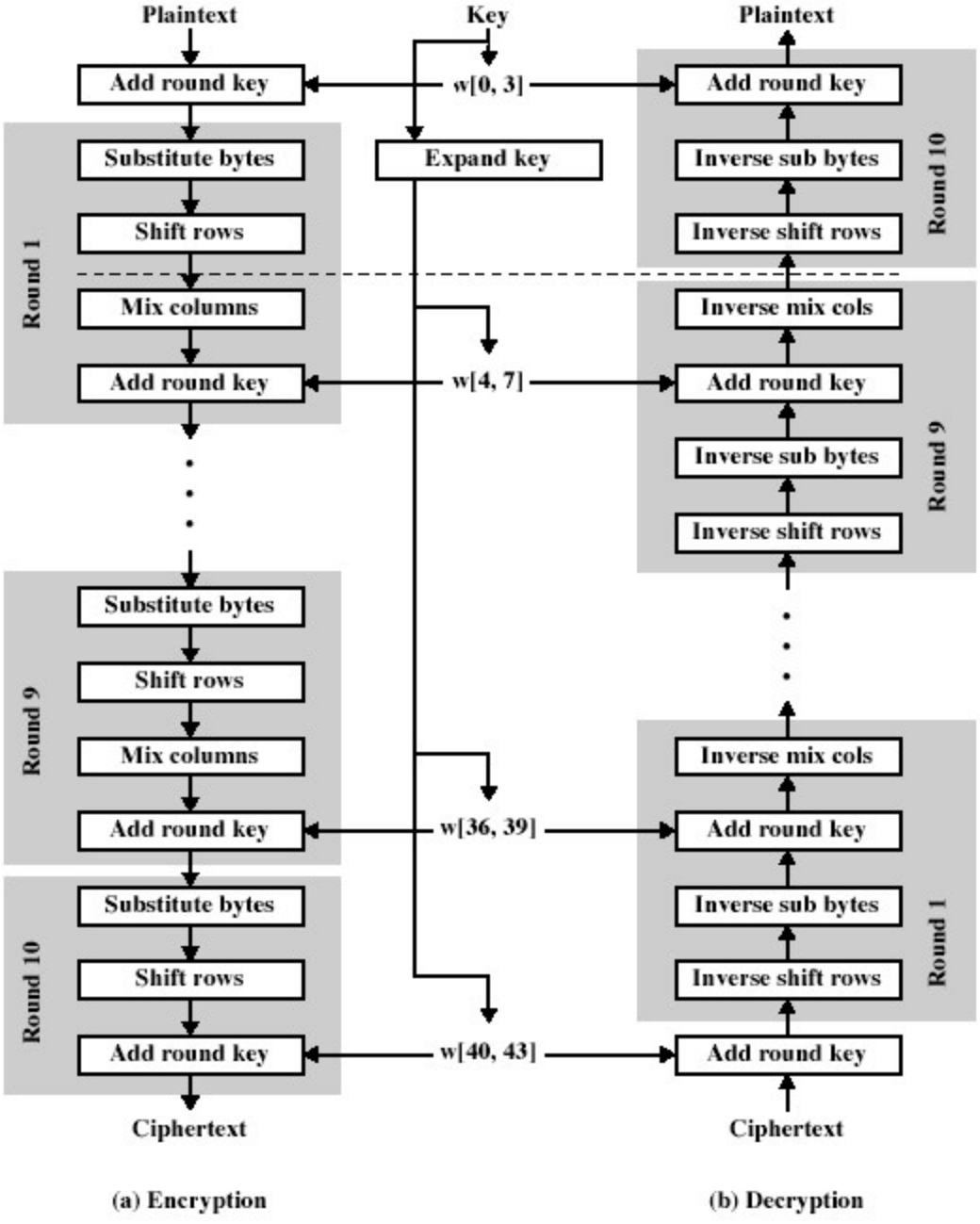
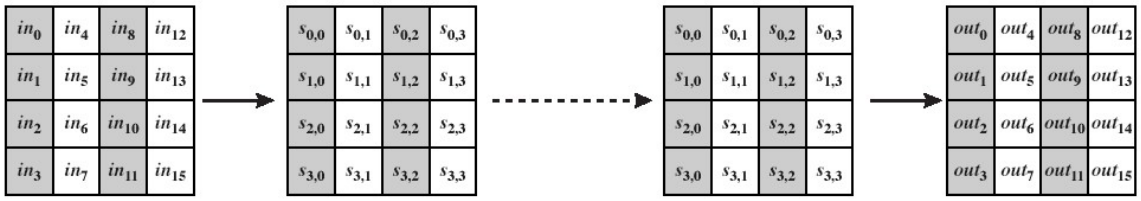


Figure 5.1 AES Encryption and Decryption

The input to the encryption and decryption algorithm is a single 128-bit block, this block, in FIPS PUB 197, is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. These operations are depicted in Figure 5.2a:

OVERALL STRUCTURE OF AES (Cont 1)



(a) Input, state array, and output

(b) Key and expanded key

Figure 5.2 AES Data Structures

Similarly, the 128-bit is depicted as a square matrix of bytes. This key is expanded into an array of key schedule words; each word is 4 bytes and the total key schedule is 44 words for the 128-bit key (Figure 5.2b). Ordering of bytes within a matrix is by column.

Before delving into details, we can make several comments about overall AES structure:

1. This cipher is not a Feistel structure.
2. The key that is provided as input is expanded into an array of 44 words (32-bits each), w[i]. 4 distinct words (128 bits) serve as a round key for each round; these are indicated in Fig. 5.1
3. 4 different stages are used, 1 permutation and 3 of substitution:

- Substitute bytes – Uses an S-box to perform a byte-to-byte substitution of the block
- Shift rows – A simple permutation
- Mix columns – A substitution that makes use of arithmetic over GF(2^8).
- Add round key – A simple bitwise XOR of the current block with the portion of the expanded key

4. The structure is quite simple. Figure 5.3 depicts the structure of a full encryption round

OVERALL STRUCTURE OF AES (Cont 2)

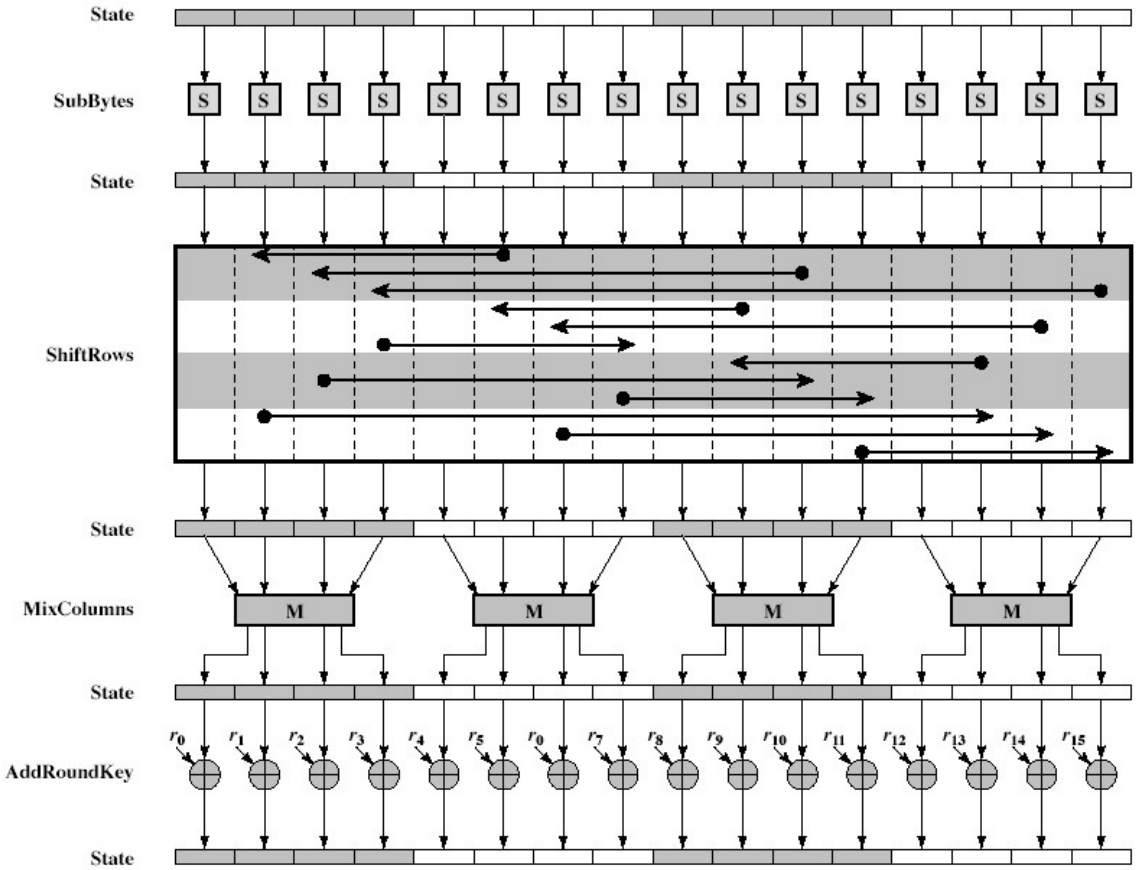


Figure 5.3 AES Encryption Round

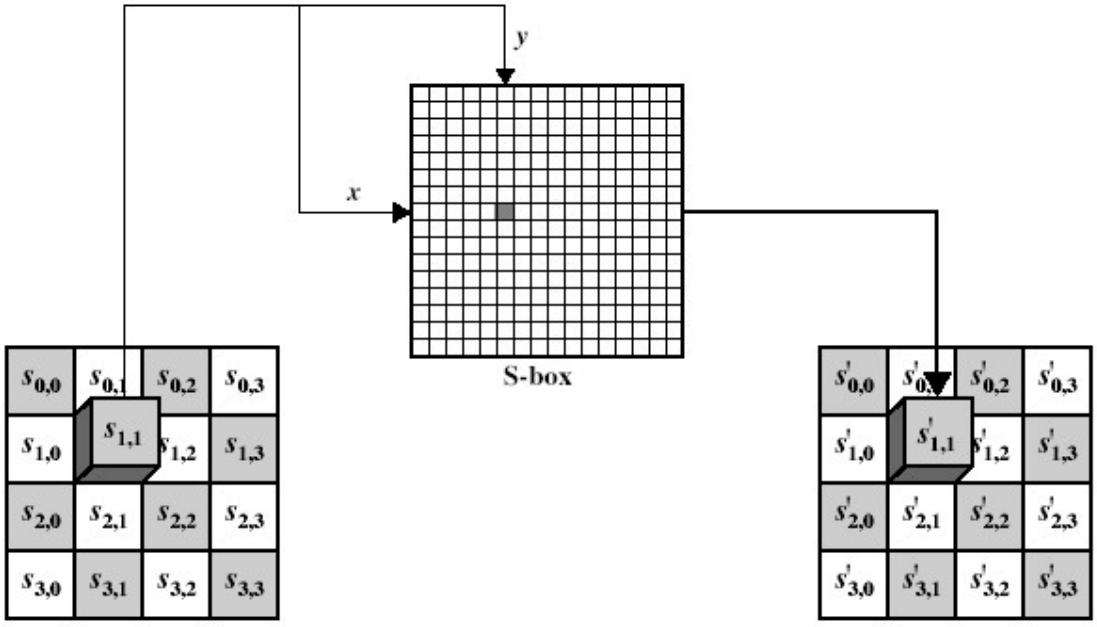
5. Only the Add Round Key stage uses the key. Any other stage is reversible without knowledge of the key.
6. The Add Round Key is a form of Vernam cipher and by itself would not be formidable. The other 3 stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (Add Round Key), followed by scrambling of the block.
7. Each stage is easily reversible
8. Decryption uses the same keys but in the reverse order. Decryption is not identical to encryption
9. At each horizontal point (e.g., the dashed line) in Figure 5.1, State is the same for both encryption and decryption
10. The final round of both encryption and decryption consists of only 3 stages; it is the consequence of the particular structure of AES.

OVERALL STRUCTURE OF AES (Cont 3)

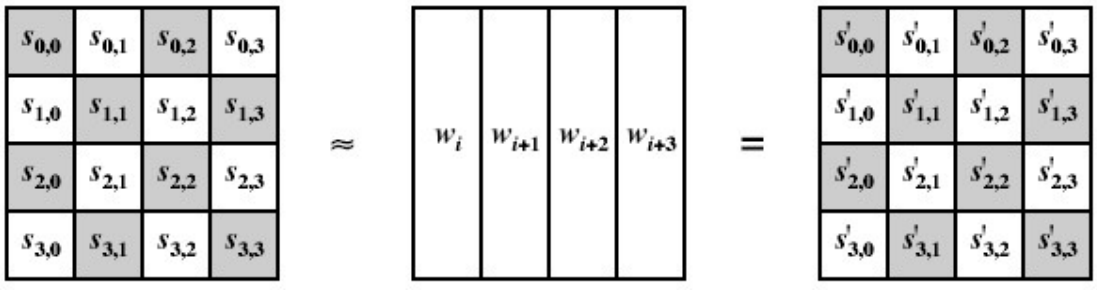
As was mentioned in Chapter 4, AES uses arithmetic in the finite field GF(2^8), with the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.

Substitute Byte Transformation. Forward and Inverse Transformation

The Forward substitute byte transformation, called SubBytes, is a simple table lookup (Figure 5.4a).



(a) Substitute byte transformation



(b) Add Round Key Transformation

Substitute Byte Transformation. Forward and Inverse Transformation (Cont 2)

AES defines a 16x16 matrix of byte values, called an S-box (Table 5.4a), that contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2a}:

Substitute Byte Transformation. Forward and Inverse Transformation (Cont 3)

Table 5.4 AES S-Boxes

(a) S-box

		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

(b) Inverse S-box

		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

The S-box is constructed in the following fashion:

1. Initialize the S-box with the byte values in ascending order row by row. Thus, the value of the byte at row x, column y is {xy}

Substitute Byte Transformation. Forward and Inverse Transformation (Cont 4)

2. Map each byte in the S-box to its multiplicative inverse in the finite field GF(2^8); the value {00} is mapped to itself.
3. Consider that each byte in the S-box consists of 8 bits labeled (b7,b6,b5,b4,b3,b2,b1,b0). Apply the following transformation to each bit of each byte in the S-box:

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (5.1)$$

where c_i is the i-th bit of byte c with the value {63}, that is, (c7c7c5c4c3c2c1c0)=(01100011). The prime $(')$ indicates that the variable is to be updated by the value on the right. The AES standard depicts this transformation in matrix form as follows:

B_0'		1	0	0	0	1	1	1	1		B_0		1	
B_1'		1	1	0	0	0	1	1	1		B_1		1	
B_2'		1	1	1	0	0	0	1	1		B_2		0	
B_3'	=	1	1	1	1	0	0	0	1	x	B_3	+	0	(5.2)
B_4'		1	1	1	1	1	0	0	0		B_4		0	
B_5'		0	1	1	1	1	1	0	0		B_5		1	
B_6'		0	0	1	1	1	1	1	0		B_6		1	
B_7'		0	0	0	1	1	1	1	1		B_7		0	

Each element in the product matrix is the bitwise XOR of elements of one row and one column. Further, the final addition, shown in (5.2), is a bitwise XOR.

As an example, consider the input value {95}. The multiplicative inverse in GF(2^8) is $\{95\}^{-1} = \{8a\}$, which is 10001010 in binary. Using equation (5.2),

1	0	0	0	1	1	1	1		0		1		1		1		0
1	1	0	0	0	1	1	1		1		1		0		1		1
1	1	1	0	0	0	1	1		0		0		0		0		0
1	1	1	1	0	0	0	1	x	1	+	0	=	1	+	0	=	1
1	1	1	1	1	0	0	0		0		0		0		0		0
0	1	1	1	1	1	0	0		0		1		0		1		1
0	0	1	1	1	1	1	0		0		1		1		1		0
0	0	0	1	1	1	1	1		1		0		0		0		0

The result is {2a}, which should appear in row {09} column {05} of the S-box. This is verified by checking Table 5.4a.