COMP3260/6360 Data Security Lecture 7

Prof Ljiljana Brankovic
School of Electrical Engineering and Computer Science

Lecture Overview

1. Origins of AES
   a) AES Requirements
   b) AES Evaluation Criteria
   c) AES Shortlist
   d) Final NIST Evaluation of Rijndael

2. Rijndael
   a) Overall Structure
   b) Details of a single round
      i. Byte Substitution
      ii. Shift Rows
      iii. Mix Columns
      iv. Add Round Key
   c) Key Expansion
   d) AES Decryption
   e) Implementation Issues

3. AES as polynomial arithmetic with coefficients in GF(2^8)

Advanced Encryption Standard

- Chapter 6, Advanced Encryption Standard

Note that in-text references and quotes are omitted for clarity of the slides. When you write an essay or report it is very important that you use both in-text references and quotes where appropriate.

Advanced Encryption Standard

"It seems very simple."

"It is very simple. But if you don't know what the key is it's virtually indecipherable."

—***Talking to Strange Men,*** **Ruth Rendell**

Origins

- Clearly, a replacement for DES was needed
  - have demonstrated exhaustive key search attacks

- Triple-DES could be used instead:
  - uses the algorithm that has been exposed to more scrutiny than any other algorithm
  - if only security was considered, 3DES would have been an appropriate choice

Origins

- 3DES has the following drawbacks:
  - DES itself was designed for mid 70s hardware implementations and does not produce efficient software code; 3DES has three times as many rounds as DES
  - uses 64 bit blocks - larger block size is needed.

Origins

- US NIST issued call for ciphers in 1997
- Out of 21 submissions 15 candidates accepted in Jun 98:
  - CAST-256 (Entrust Technologies)
  - CRYPTON (Future Systems)
  - DEAL (Richard Outerbridge, Lars Knudsen)
  - DFC (National Centre for Scientific Research, France)
  - E2 (NTT)
  - FROG (TecApro Internacional)
  - HPC (Rich Schroeppel)
  - **LOKI97 (Lawrie Brown, Josef Pieprzyk, Jenniffer Seberry)**
  - MAGENTA (Deutche Telekom)
  - Mars (IBM)
  - RC6 (RSA)
  - **Rijndael (Joan Daemon, Vincent Rijmen)**
  - Safer+ (Cylink)
  - Serpant (Ross Anderson, Eli Biham, Lars Knudsen)
  - Twofish (Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson).

Origins

- 5 were shortlisted in August 1999.
  - MARS
  - RC6
  - Rijndael
  - Serpent
  - Twofish

- Rijndael was selected as the AES in October 2000 and issued as FIPS PUB 197 standard in November 2001.

AES Requirements

- secret key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- faster than Triple-DES
- active life of 20-30 years (+ archival use)
- provide full specification & design details
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- Initial criteria:
  - security – effort to practically cryptanalyse
  - cost
  - algorithm & implementation characteristics

AES Initial Evaluation Criteria

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.

AES Initial Evaluation Criteria

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.

AES Initial Evaluation Criteria

COST

- **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.

AES Initial Evaluation Criteria

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.).

AES Initial Evaluation Criteria

ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

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.

AES Evaluation Criteria

- Final criteria
  - general security
  - software & hardware implementation ease
  - restricted space environments
  - implementation attacks
  - encryption vs decryption
  - key agility
  - flexibility (other key and block sizes, increasing number of rounds, etc)
  - potential for instruction-level parallelism

AES Shortlist

- after testing and evaluation, shortlist in Aug-99:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
- then subject to further analysis & comment
- saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - refined existing ciphers verses new proposals

The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - treats data in 4 groups of 4 bytes
  - operates an entire block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

Software Implementations

- Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's highly 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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Final NIST Evaluation of Rijndael (October 2, 2000)

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.

Rijndael (AES)

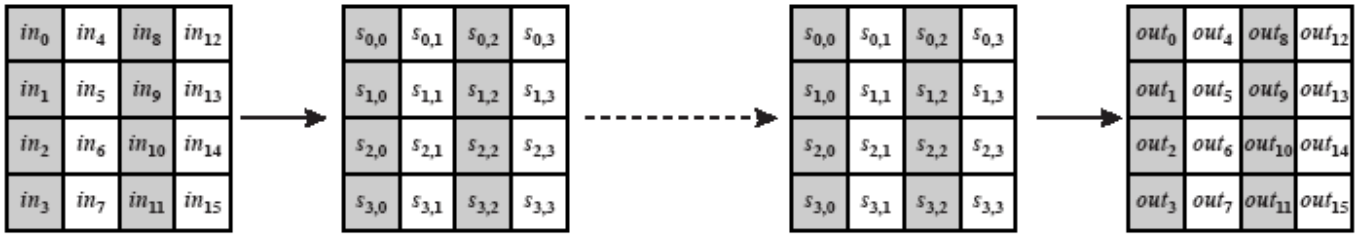
- processes data as 4 groups of 4 bytes (state)
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiplication of groups)
  - add round key (XOR state with key material)
- initial XOR key material & incomplete last round (10^{th}/12^{th}/14^{th})
- all operations can be combined into XOR and table lookups - hence very fast & efficient

Rijndael (AES)

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

Rijndael (AES)

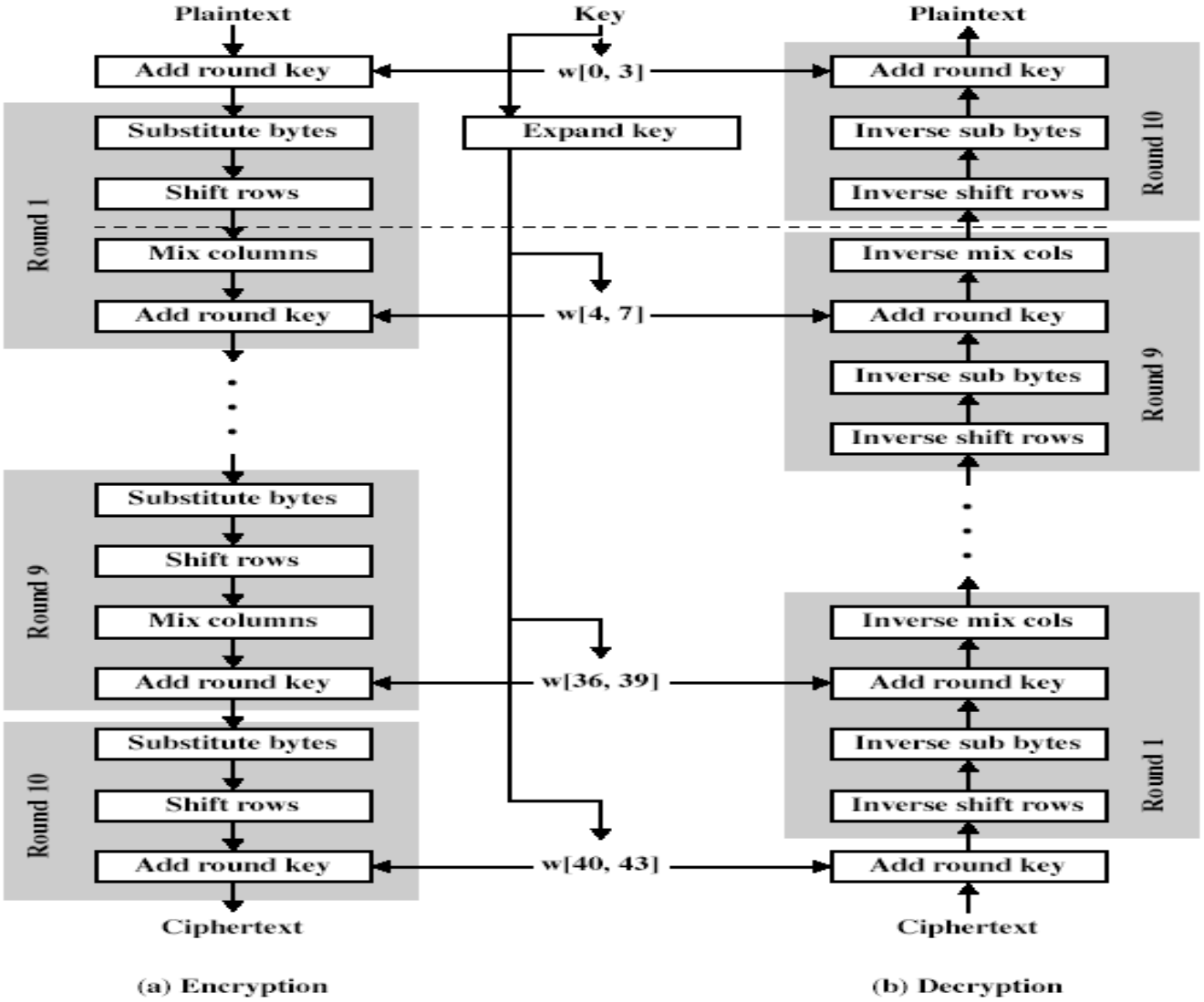


(a) Input, state array, and output

(b) Key and expanded key

Figure 5.2 AES Data Structures

Rijndael (AES)



Rijndael (AES)

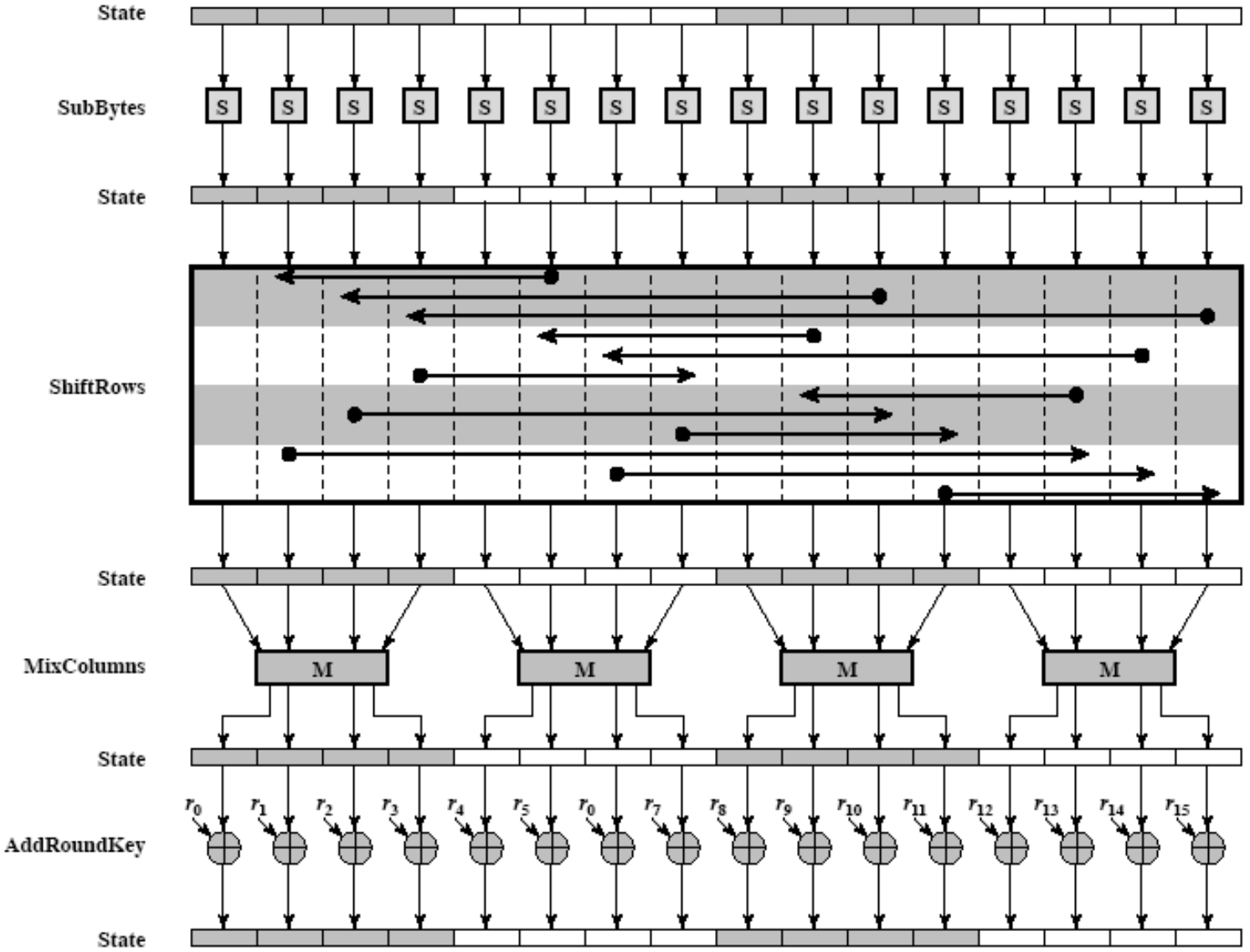
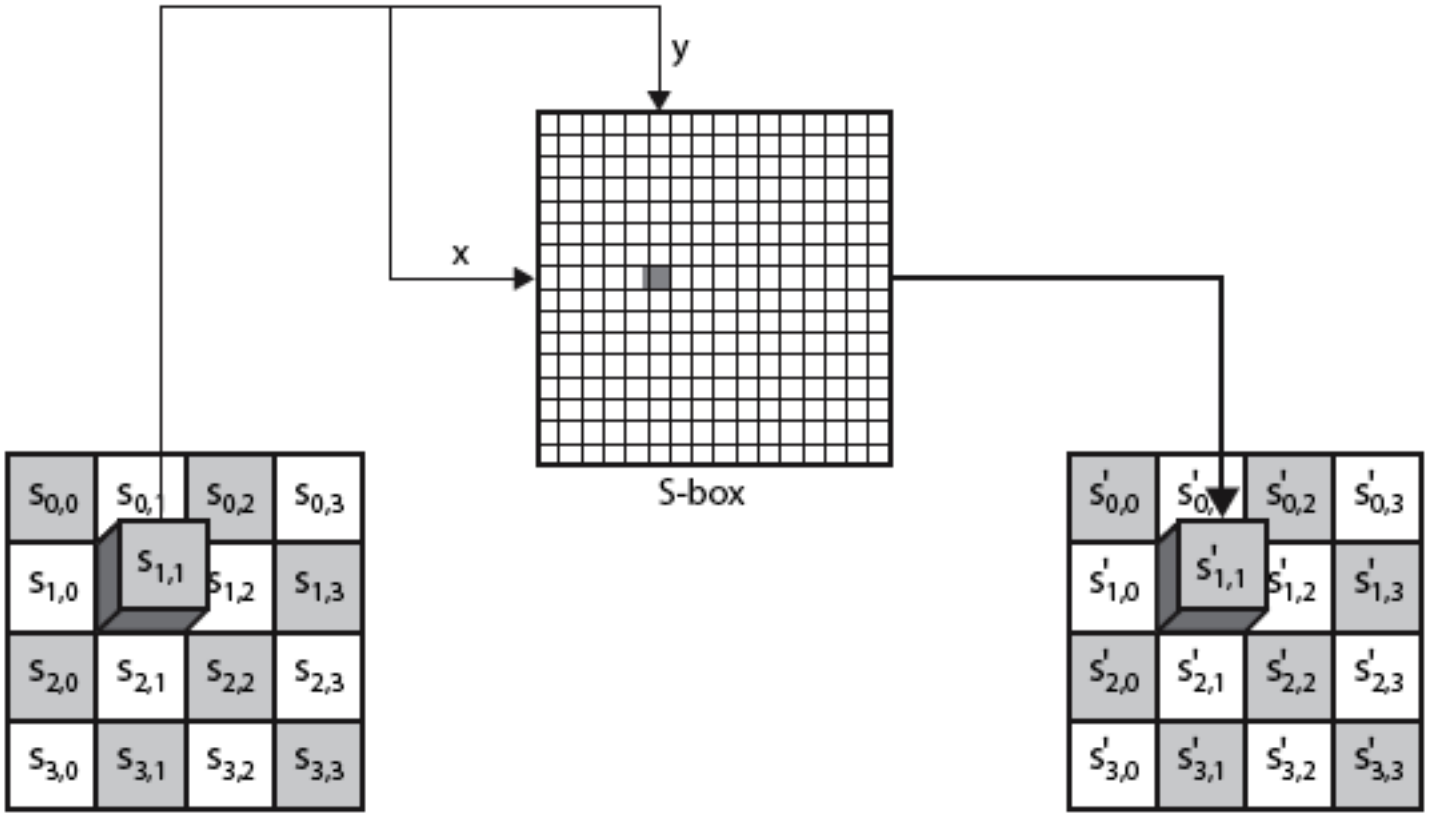


Figure 5.3 AES Encryption Round

Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by row 9 col 5 byte
  - which is the value {2A}
- S-box is constructed using a defined transformation of the values in $GF(2^8)$
- designed to be resistant to all known attacks

Byte Substitution



Byte Substitution

Table 5.4 AES S-Boxes

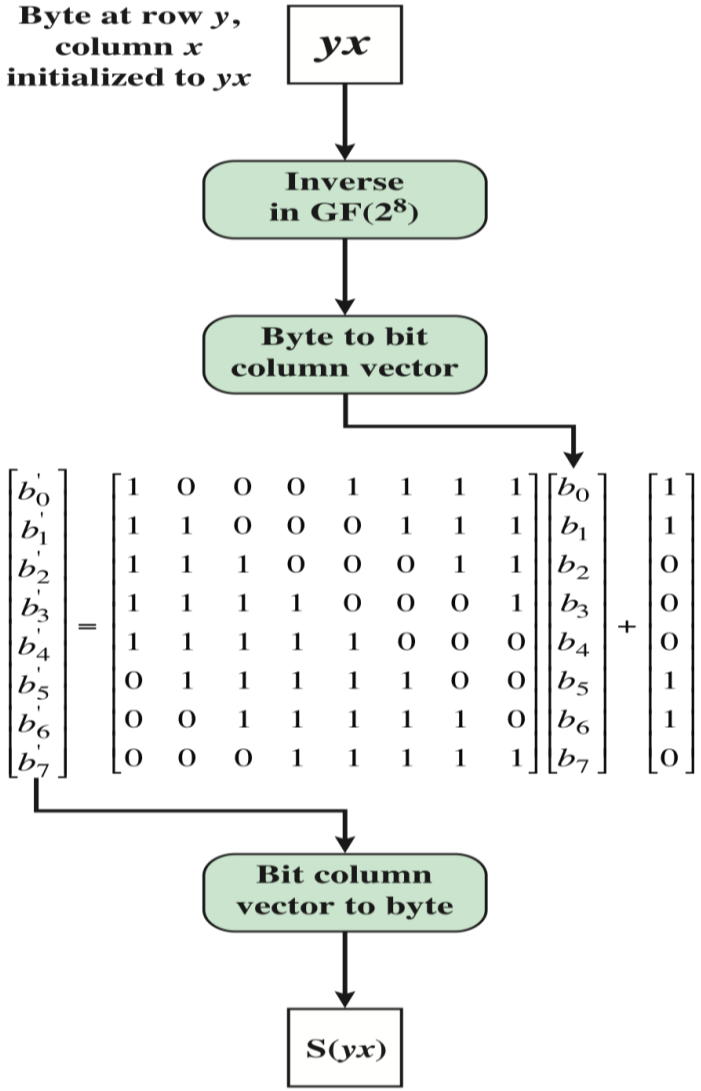
(a) S-box

		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

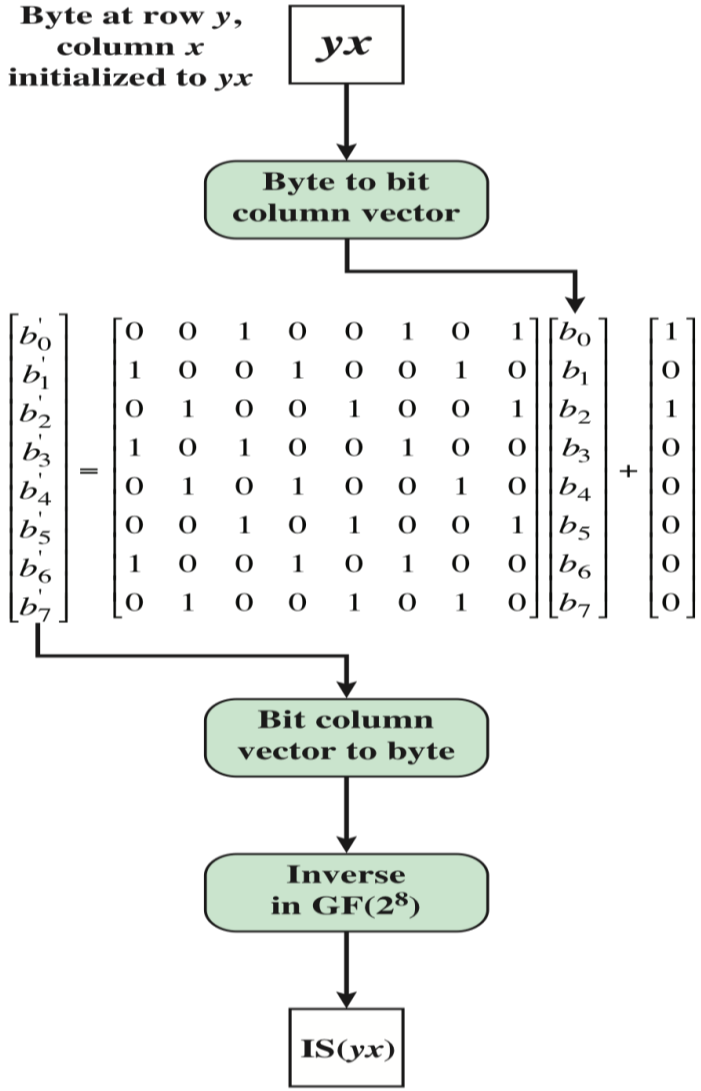
Byte Substitution

(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



(a) Calculation of byte at row y, column x of S-box



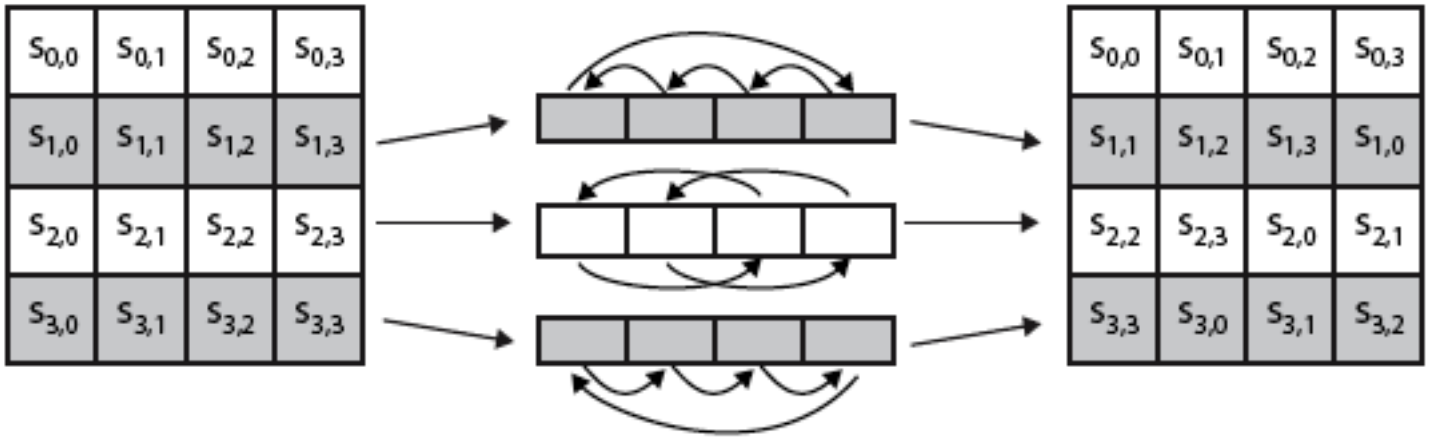
(a) Calculation of byte at row y, column x of IS-box

Figure 5.6 Construction of S-Box and IS-Box

Shift Rows

- a circular byte shift in each row
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt does shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Shift Rows


$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

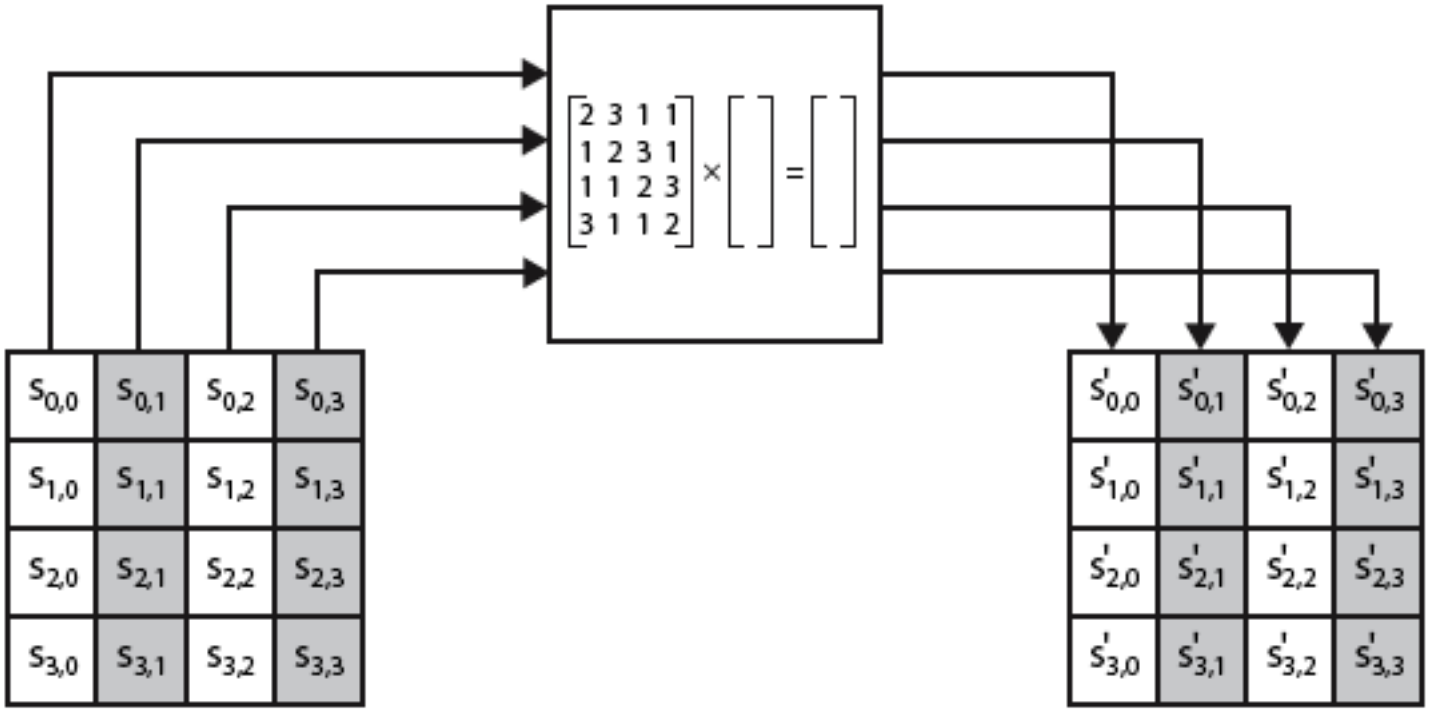
Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using irreducible polynomial

$$m(x) = x^8+x^4+x^3+x+1$$

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

Mix Columns


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

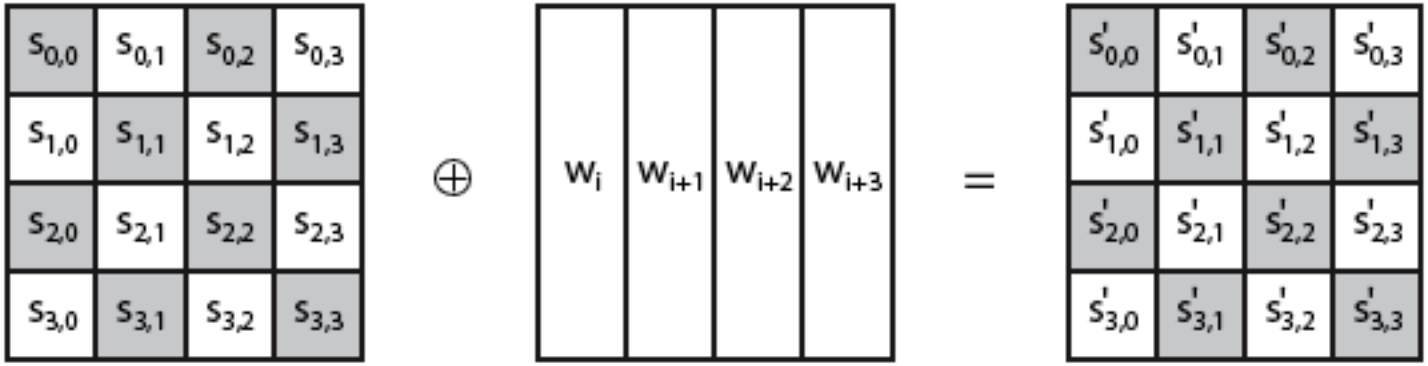
Mix Columns

- can express each col as 4 equations
  - to derive each new byte in col
- decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- have an alternate characterisation
  - each column a 4-term polynomial
  - with coefficients in GF(2^8)
  - and polynomials multiplied modulo (x^4+1)

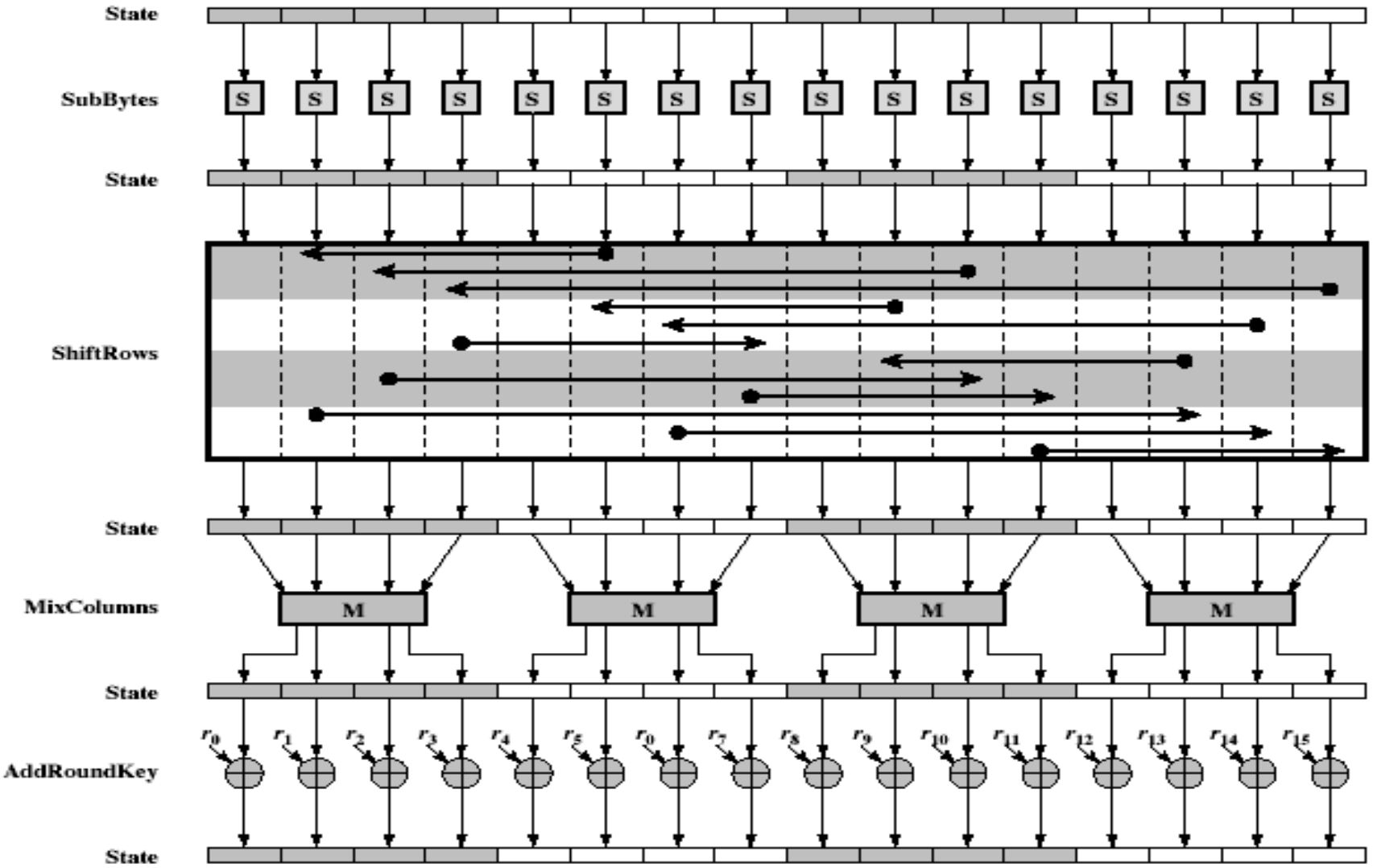
Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security

Add Round Key



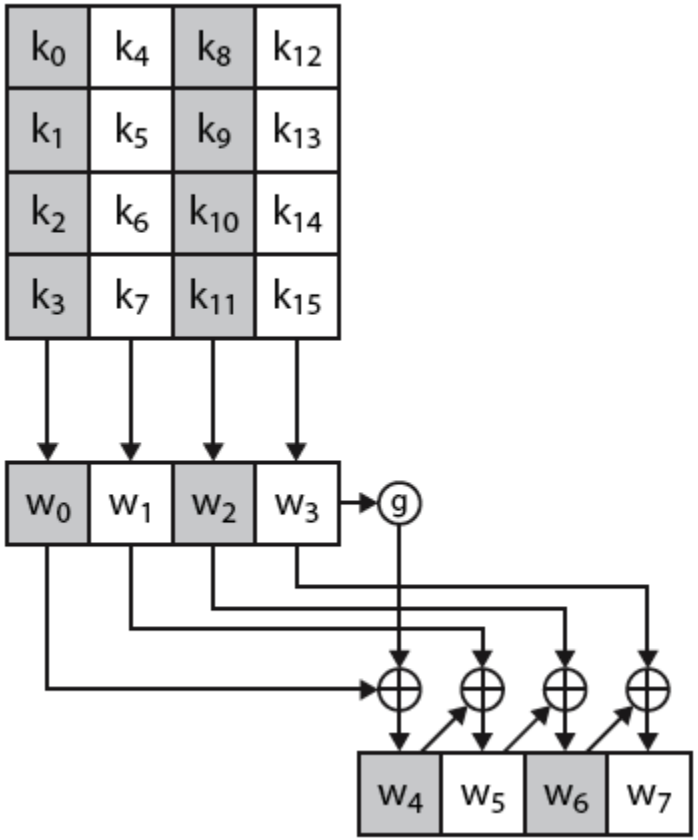
AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
State

AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - every 4th has S-box + rotate + XOR round constant on previous before XOR together
- designed to resist known attacks

AES Key Expansion



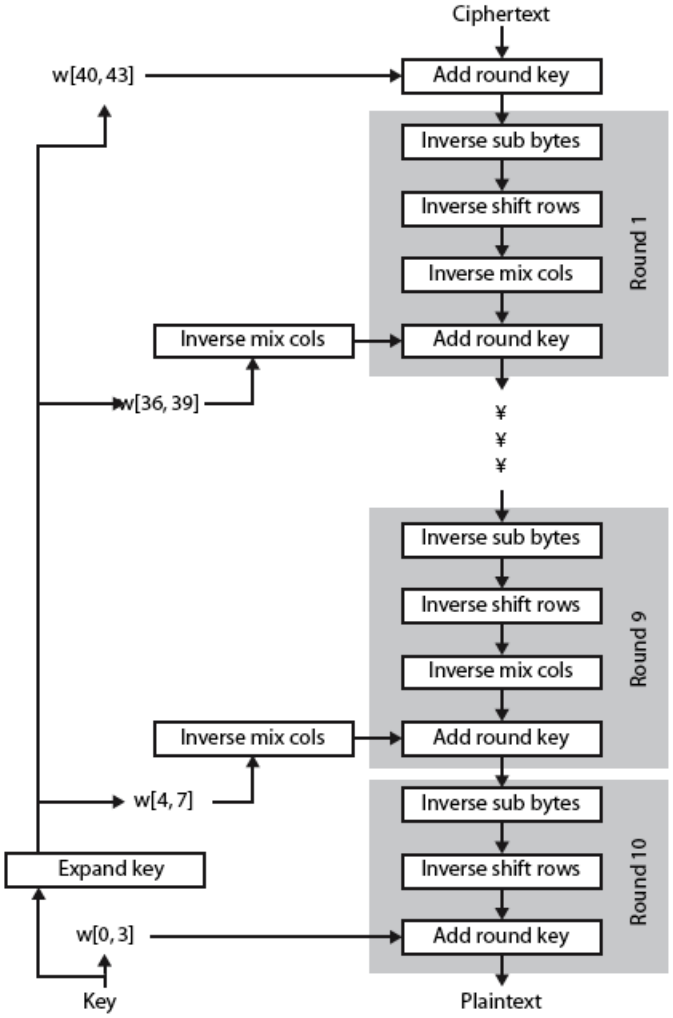
Key Expansion Rationale

- designed to resist known attacks
- design criteria included
  - knowing part key insufficient to find many more
  - invertible transformation
  - fast on wide range of CPU's
  - use round constants to break symmetry
  - diffuse key bits into round keys
  - enough non-linearity to hinder analysis
  - simplicity of description

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

AES Decryption



Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shift
  - add round key works on byte XOR's
  - mix columns requires matrix multiply in GF(2^8) which works on byte values, can be simplified to use table lookups & byte XOR's

Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 4Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

AES: polynomial with coefficients in $GF(2^8)$

- Arithmetic of AES can be thought of as polynomial arithmetic for polynomials of degree at most 3 with coefficient in $GF(2^8)$.

- Recall that the **state** is a 4 x 4 matrix where each cell is a byte; the bytes are originally entered into the state column by column.

- Each column of the state can be thought of as polynomial of degree up to 3, where the 4 bytes are the 4 coefficients; each coefficient can be thought of as a polynomial in $GF(2^8)$.

AES: polynomial with coefficients in $GF(2^8)$

- Addition of polynomials with degree up to 3 with coefficients in $GF(2^8)$:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$
$$b(x) = b_3x^3 + b_2x^2 + b_1x + a_0$$

$$a(x) + b(x) = (a_3 \oplus b_3)\,x^3 + (a_2 \oplus b_2)\,x^2 + (a_1 \oplus b_1)\,x + (a_0 \oplus b_0)$$

AES: polynomial with coefficients in $GF(2^8)$

- Multiplication of polynomials with degree up to 3 with coefficients in $GF(2^8)$: coefficients are multiplied in $GF(2^8)$ and the result is reduced $\mod (x^4 + 1)$

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$
$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$c(x) = a(x) \times b(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$
$$d(x) = c(x) \mod (x^4 + 1)$$

AES: polynomial with coefficients in $GF(2^8)$

$$c(x)=a(x)\times b(x)=c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0$$

$$c_0=a_0b_0$$
$$c_1=a_1b_0\oplus a_0b_1$$
$$c_2=a_2b_0\oplus a_1b_1\oplus a_0b_2$$
$$c_3=a_3b_0\oplus a_2b_1\oplus a_1b_2\oplus a_0b_3$$
$$c_4=a_3b_1\oplus a_2b_2\oplus a_1b_3$$
$$c_5=a_3b_2\oplus a_2b_3$$
$$c_6=a_3b_3$$

AES: polynomial with coefficients in $GF(2^8)$

$d(x) = c(x) \bmod (x^4+1)$

To calculate $d(x)$ we use the following observation:

$$x^i \bmod (x^4+1) = x^{i \bmod 4}$$

We then have

$$\begin{aligned} d(x) &= c(x) \bmod (x^4+1) \\ &= (c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0) \bmod (x^4+1) \\ &= c_3x^3+(c_2 \oplus c_6)x^2+(c_1 \oplus c_5)x+(c_0 \oplus c_4) \end{aligned}$$

AES: polynomial with coefficients in $GF(2^8)$

The coefficients of $d(x)$ are as follows:

$$d_0 = a_0 b_0 \oplus a_3 b_1 \oplus a_2 b_2 \oplus a_1 b_3$$
$$d_1 = a_1 b_0 \oplus a_0 b_1 \oplus a_3 b_2 \oplus a_2 b_3$$
$$d_2 = a_2 b_0 \oplus a_1 b_1 \oplus a_0 b_2 \oplus a_3 b_3$$
$$d_3 = a_3 b_0 \oplus a_2 b_1 \oplus a_1 b_2 \oplus a_0 b_3$$

AES: polynomial with coefficients in $GF(2^8)$

The coefficients of $d(x)$ can also be written in matrix form:

$$\begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{vmatrix} = \begin{vmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{vmatrix} \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{vmatrix}$$

AES: polynomial with coefficients in $GF(2^8)$

Recall that the MixColumns transformation in AES was defined as a multiplication of a fixed matrix and the state:

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

AES: polynomial with coefficients in $GF(2^8)$

Another way to think about MixColumns is to see each columns of the State matrix as a four-term polynomial with coefficients in **$GF(2^8)$**; then each column is multiplied by a fixed polynomial $a(x) \bmod (x^4+1)$, where

$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$,

$a_3=03$, $a_2=01$, $a_1=01$, $a_0=02$ (in hexadecimal)

$$\begin{vmatrix} s'_{0,0} \\ s'_{1,0} \\ s'_{2,0} \\ s'_{3,0} \end{vmatrix} = \begin{vmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{vmatrix} \begin{vmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{vmatrix}$$

Next week:

1. Public-Key Cryptography

2. RSA
   a) The underlying mathematics
   b) Security of RSA

3. ElGamal Cryptography

4. Diffie-Hellman

Chapter 9 Public Key Cryptography and RSA
Chapter 10, section 10.2 ElGamal Cryptographic System
Chapter 14, Diffie-Hellman key exchange
Original paper on RSA by Rivest, Shamir and Adleman
Key Exchange

References

1. W. Stallings. "Cryptography and Network Security", Global edition, Pearson Education Australia, 2016.

2. Official textbook slides by L. Brown.