Number Theory and Cryptography

Mohona Ghosh

Motivation for Advanced Encryption Standard (AES)

- Luke O' Conners (IBM):

  Most Ciphers are secure after sufficiently many rounds.

- James L. Massey (ETH, Zurich):

  Most ciphers are too slow after sufficiently many rounds.

Cipher Design: Science or Engineering ?

- Practical security can be achieved easily if we don't worry about performance.
- It is not sufficient to prove that a secure block cipher exists, we also have to construct it.
- Design challenges:
  1. Security and performance.

Motivation for Advanced Encryption Standard (AES)

- Slow speed of 3-DES on software
  - DES was designed for efficient h/w implementation not s/w
  - 3-DES – has 3 times as many rounds as DES

- Smaller block size of DES and 3-DES (64-bits)
  - Prone to few cryptanalytic attacks such as birthday attack
- In 1997, NIST (National Institute of Standards and Technology) issued a call for a new proposal
  - Security strength $\geq$ Triple DES
  - Block length of 128-bits

AES Competition

- Aug 1998: 15 submissions were accepted
- Aug 1999: 5 finalists were chosen
- Nov 2001: Rijndael selected as the winner

  Designed by Belgian researchers Joan Daemen and Vincent Rijmen

Votes after 2nd AES conference:

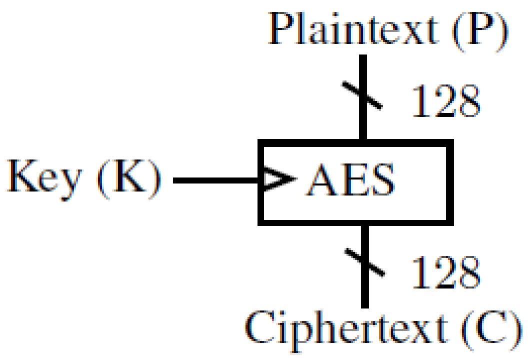
Rijndael:	86	10
Serpent:	59	7
Twofish:	31	21
RC6:	23	37
MARS:	13	84

Announcement of finalists: Aug 20,1999

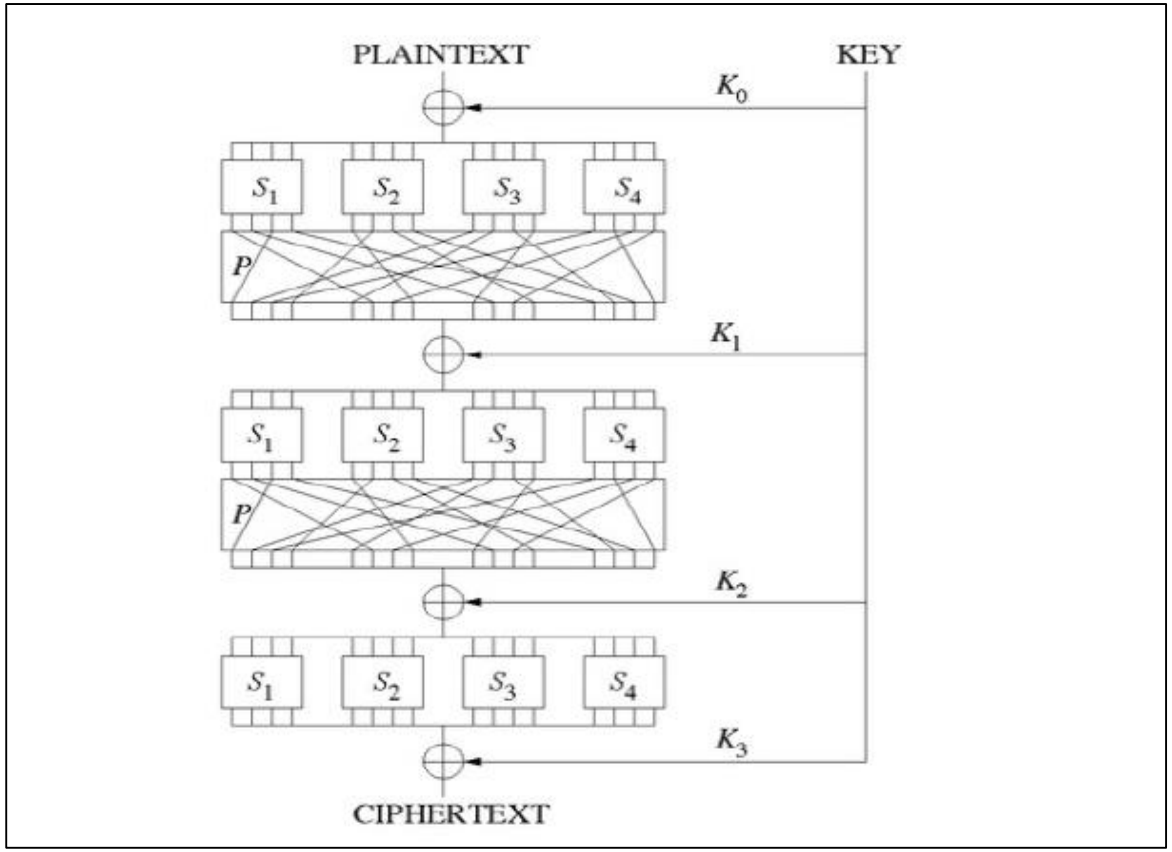
Submission	Authors
Rijndael	Joan Daemen and Vincent Rijmen
Serpent	Ross Anderson, Eli Biham and Lars Knudsen.
Twofish	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Fergusson.
	Extended team included: Stefan Lucks, Tadayoshi Kohno and Mike Stay.
RC6	Ron Rivest, Matt Robshaw, Ray Sidney and Yiqun Lisa Yin.
MARS	IBM (included Don Coppersmith).

AES

- AES is based on SPN structure
  - Substitution Permutation Network

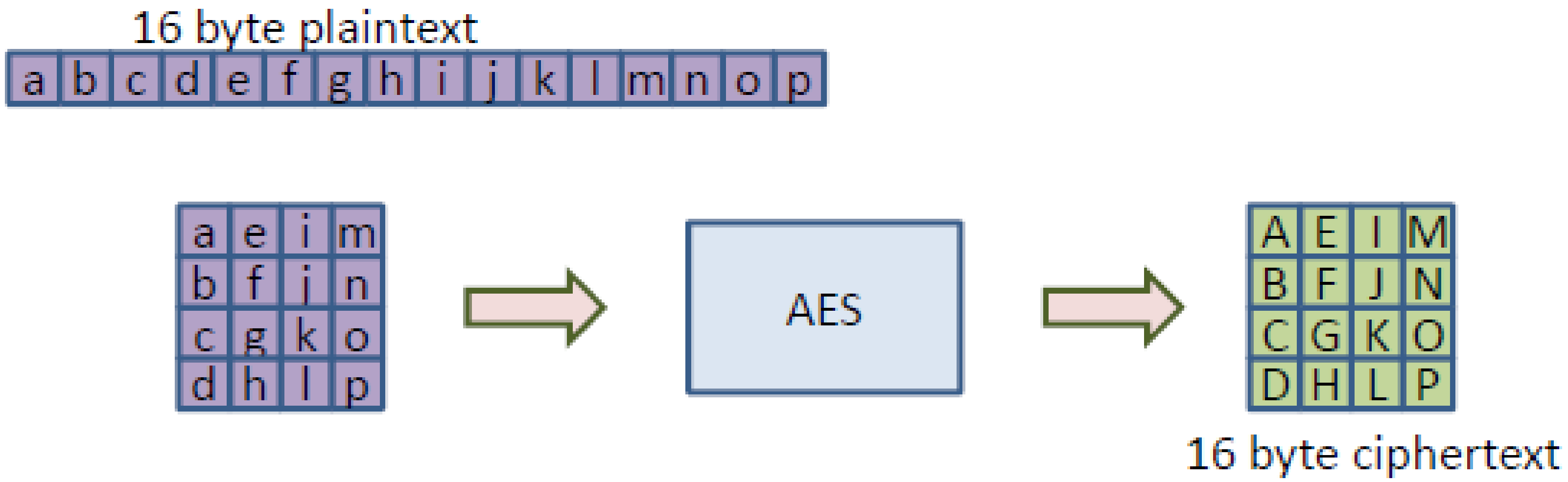


Algorithm	KeySize (in bits)	Rounds
AES-128	128	10
AES-192	192	12
AES-256	256	14



The AES state representation

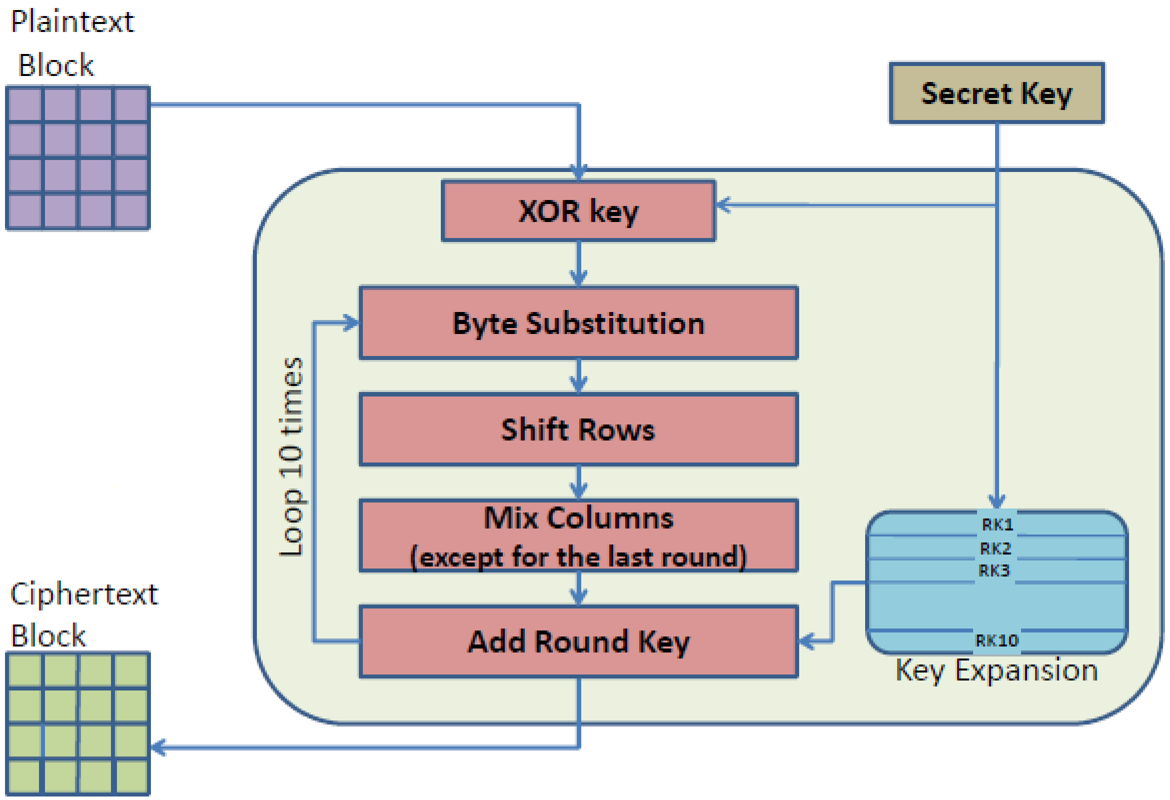
- 16 bytes (128-bits) are arranged in a 4 x 4 array as follows



AES round function

Consists of 4 steps

- SubBytes (Substitution)
- ShiftRows (Permutation)
- MixColumns (Permutation)
- AddRoundKey (Adding randomness)

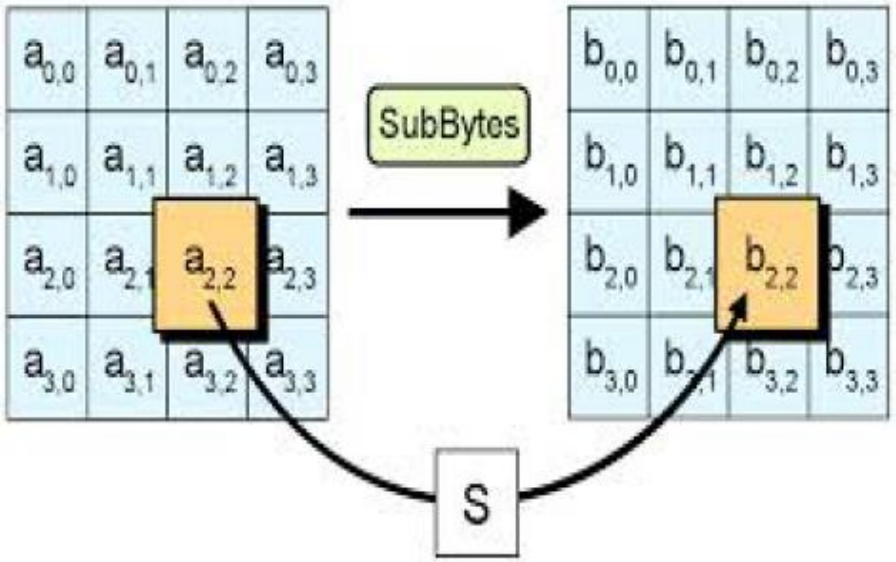


- First r-1 rounds have all the above steps
- Last r^{th} round does not have the MixColumn step
  - SubBytes, ShiftRows, AddRoundKey

AES round function

Substitution

- Makes a non linear substitution for every byte in the 4x4 matrix



AES round function

Substitution

- Makes a non linear substitution for every byte in the 4x4 matrix
- The S-box table is a as shown in the right:
  - E.g., Sbox[0x 42] = [0x 2c]
  - Use the leftmost 4 bits to select the row
  - Use the rightmost 4 bits to select the column

		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

AES round function

Substitution

- The S-box is constructed as follows:

$$For\ each\ x \in GF(2^8), \qquad x \rightarrow Ax^{-1} + b. \qquad Define\ 0^{-1} = 0.$$

- x^{-1} provides high degree of non linearity
- Provides good resistance against variety of attacks
- Affine transformation (Ax + b form) ensures that there are no fixed points, i.e., $S(x) \neq x$
    - Complicates algebraic attack
    - No opposite fixed points as well, i.e., $S(x) \neq Complement\ (x)$

$$S(x) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} x^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

A

b

AES round function

ShiftRows

- Leave the first row as it is
- Left rotate second row by 1-byte
- Left rotate third row by 2-bytes
- Left rotate fourth row by 3-bytes

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}

$\xrightarrow{\text{ShiftRow}}$

x_0	x_1	x_2	x_3
x_5	x_6	x_7	x_4
x_{10}	x_{11}	x_8	x_9
x_{15}	x_{12}	x_{13}	x_{14}

AES round function

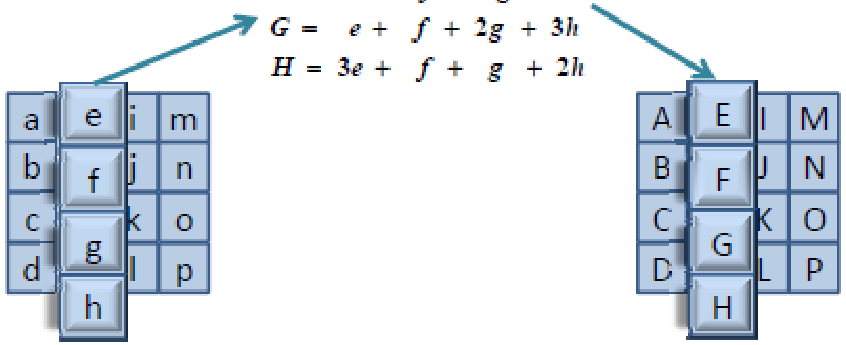
MixColumns

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

- The 4x4 state matrix is multiplied with the following matrix columnwise
- i.e., Multiplications are done in GF(2^8) as follows:
    - Irreducible polynomial ($x^8 + x^4 + x^3 + x + 1$)

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}$$

$$\begin{aligned} E &= 2e + 3f + g + h \\ F &= e + 2f + 3g + h \\ G &= e + f + 2g + 3h \\ H &= 3e + f + g + 2h \end{aligned}$$



- This step along with *ShiftRows* provides high diffusion
    - Flipping of bits in one byte in first round results in flipping of bits in all 16 bytes after 2 rounds, i.e., full diffusion is achieved

Multiplication Example

- *Let f(x) =* $x^6 + x^4 + x^2 + x + 1$ $\qquad g(x) = x^7 + x + 1$
- *Irr. Poly. m(x) =* $x^8 + x^4 + x^3 + x + 1$

$f(x)$ x $g(x) = x^{13} + x^{11} + x^9 + x^8 + x^7 +$

$x^7 + x^5 + x^3 + x^2 + x +$

$x^6 + x^4 + x^2 + x + 1$

$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

$$
\begin{array}{r l}
 & x^5 + x^3 \\
x^8 + x^4 + x^3 + x + 1 \big/ & x^{13} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + \quad + 1 \\
 & \underline{x^{13} \quad + x^9 + x^8 \quad + x^6 + x^5} \\
 & \quad x^{11} \quad + x^4 + x^3 \\
 & \quad \underline{x^{11} \quad + x^7 + x^6 \quad + x^4 + x^3} \\
 & \qquad x^7 + x^6 \qquad + 1
\end{array}
$$

Therefore, $f(x)$ x $g(x)$ mod $m(x) = x^7 + x^6 + 1$.

AES round function

Why choose this MixColumns matrix ?

- It is an MDS matrix (Maximum Distance Separable Code matrix)
  - If the input of a column changes, then all outputs change
  - This maximizes the branch number
  - Branch number of AES is 5

- Values [2,3,1,1] are the smallest which result in a circulant MDS matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

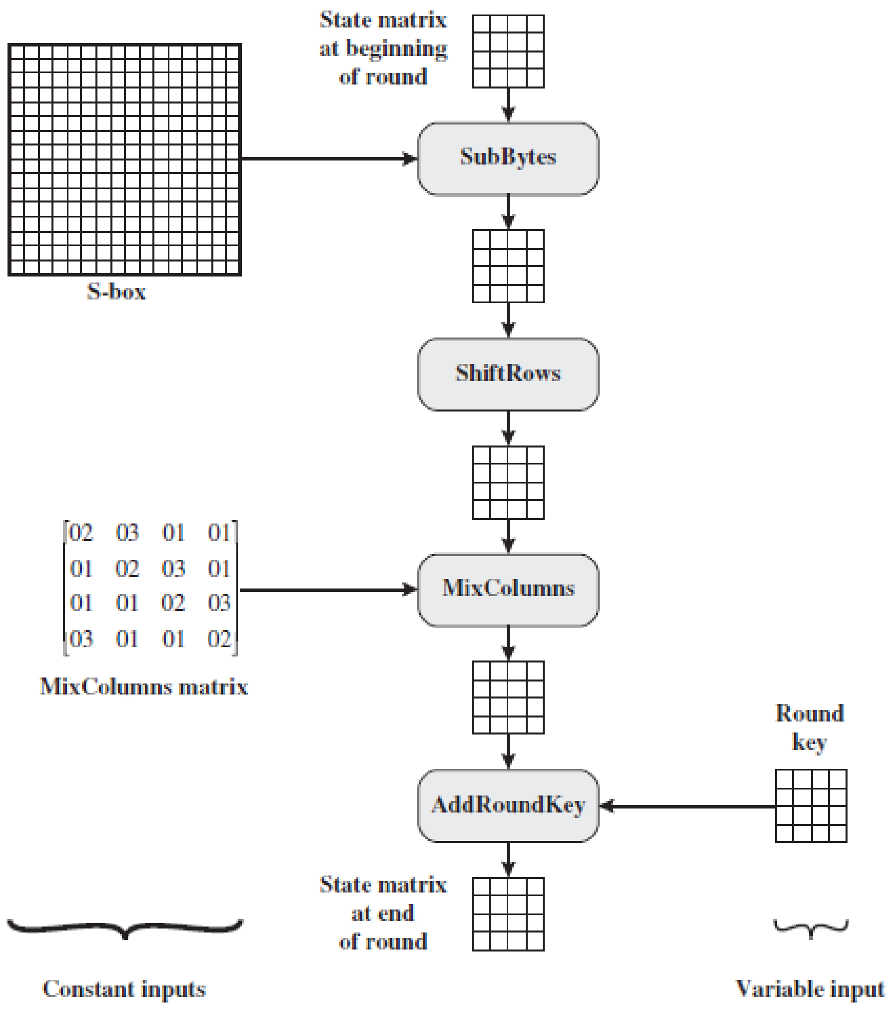
AES round function

AddRoundKey

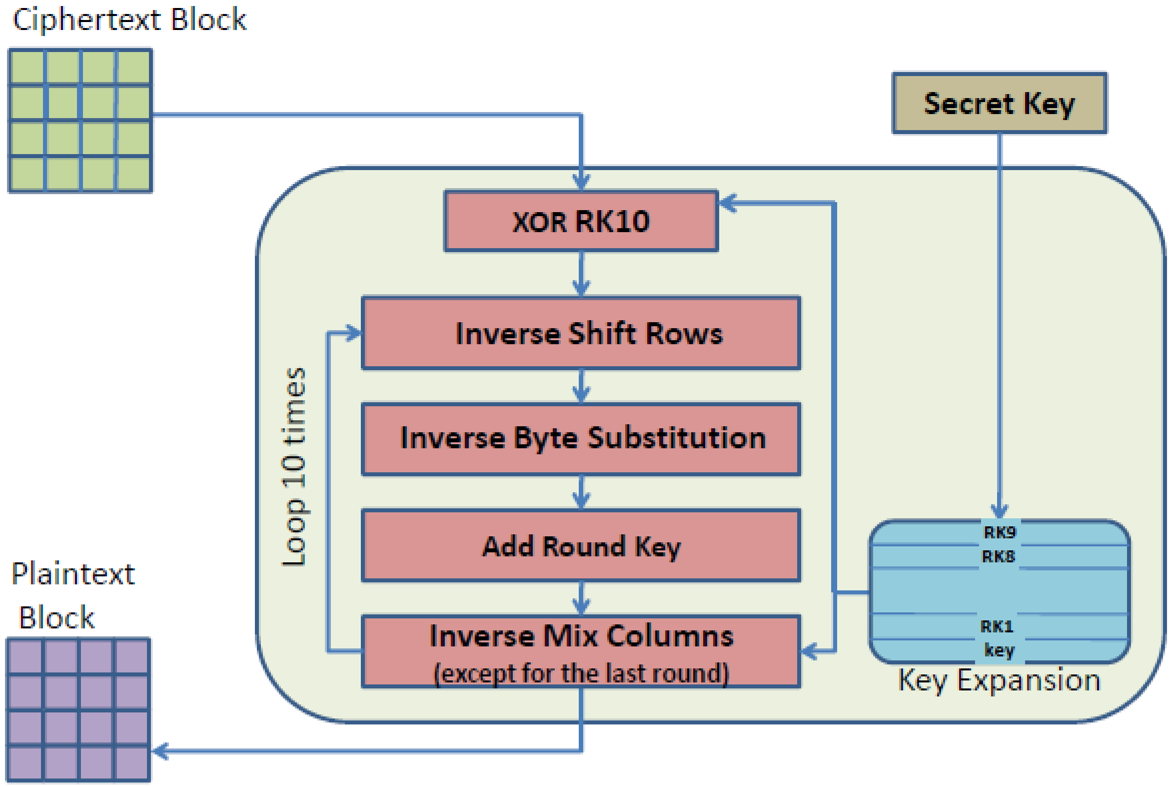
- Xor the round key to the state obtained after MixColumns operation

$$
\begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}
\oplus
\begin{bmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{bmatrix}
\Rightarrow
\begin{bmatrix} a+k_0 & e+k_4 & i+k_8 & m+k_{12} \\ b+k_1 & f+k_5 & j+k_9 & n+k_{13} \\ c+k_2 & g+k_6 & k+k_{10} & o+k_{14} \\ d+k_3 & h+k_7 & l+k_{11} & p+k_{15} \end{bmatrix}
$$

AES One Round Encryption



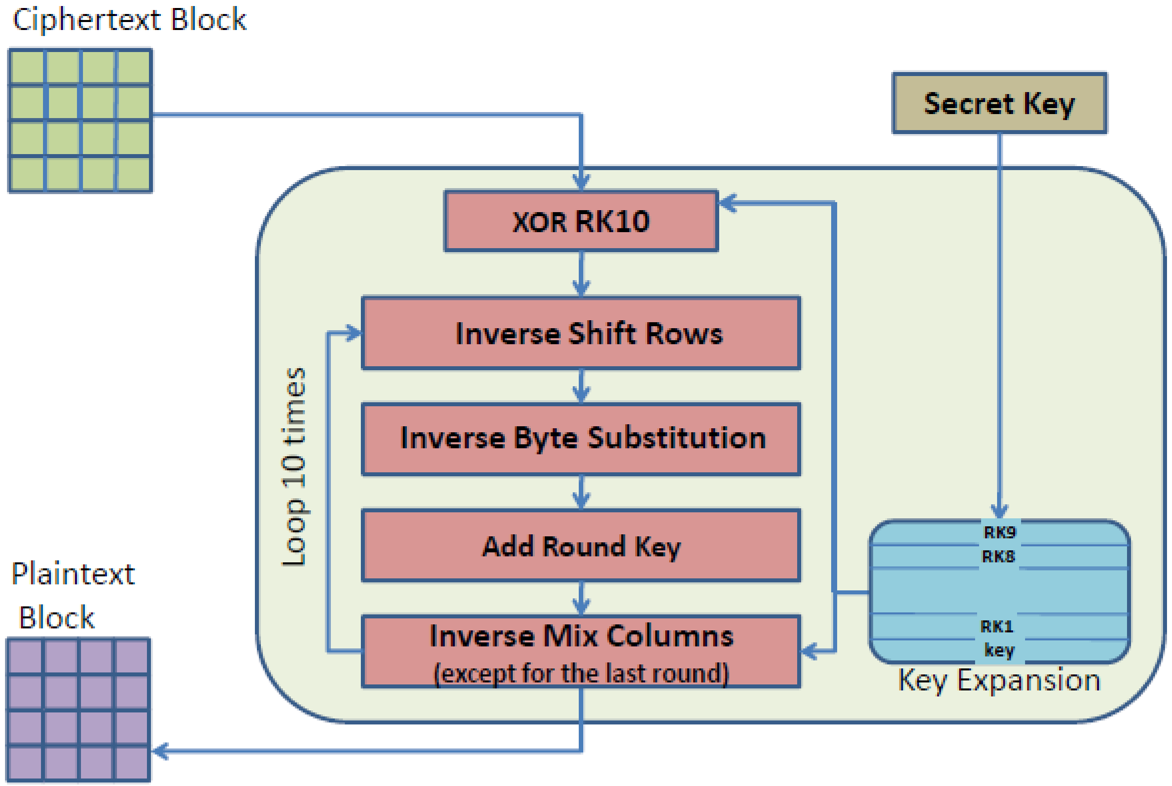
AES decryption



- Inverse Shift Rows
  - Perform circular rotations to the right
- Inverse SubBytes
  - Use the following InvSBox table

		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

AES decryption



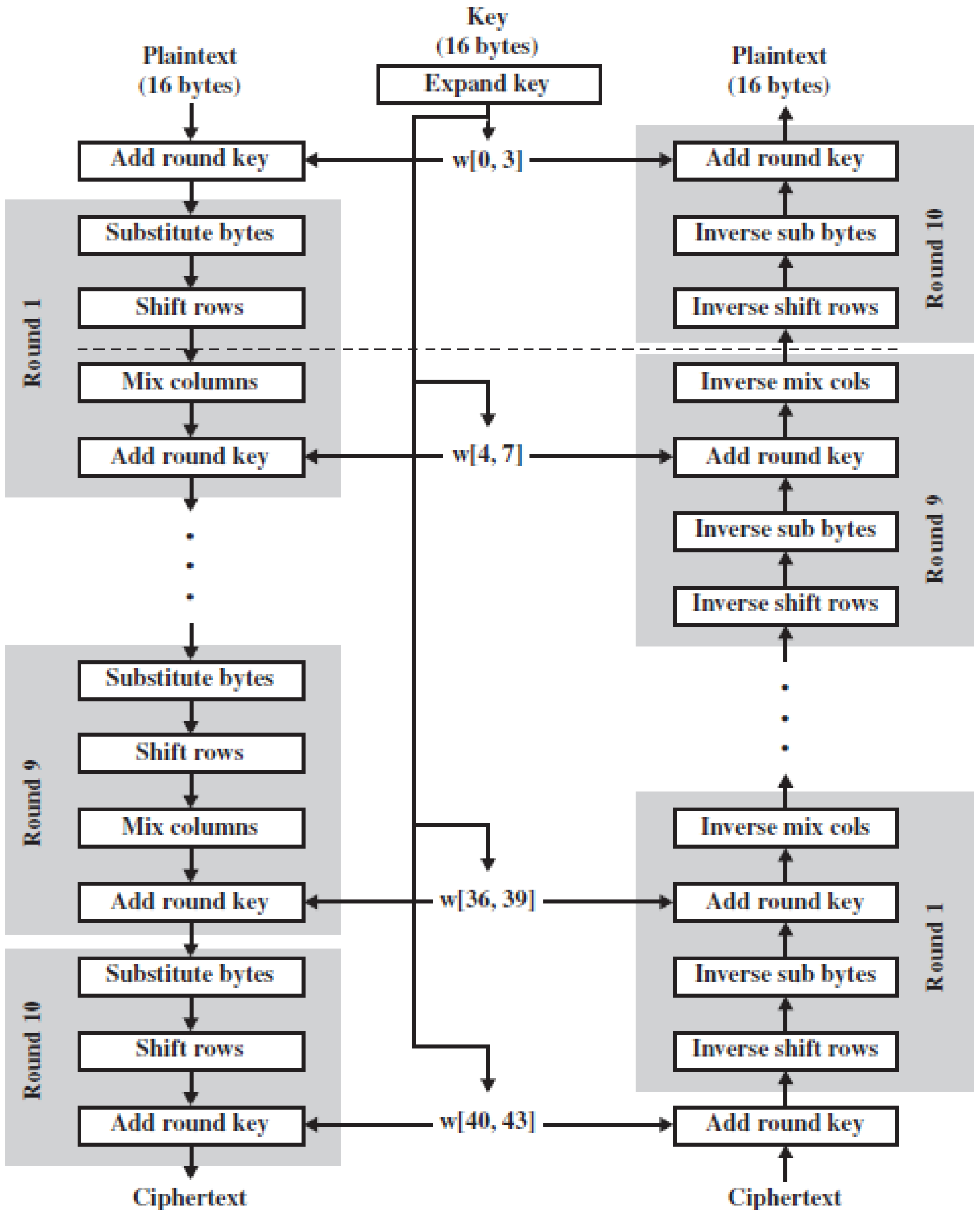
- AddRoundKey
  - Xor with round key
- Inverse MixColumns
  - Multiplication in GF(2^8) with the following matrix

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

- The round keys used in encryption and decryption process are the same
  - But in the reverse order

AES decryption

- AES decryption cipher is not identical to the encryption cipher
  - Sequence of transformations differ
- Therefore, two separate s/w or h/w is needed to support both enc. and dec.



Same Algorithm for both Enc. and Dec.

2 Round Encryption	2 Round Decryption
AddRoundKey[0]	AddRoundKey[2]
SubBytes	InvShiftRows
ShiftRows	InvSubBytes
MixColumns	AddRoundKey[1]
AddRoundKey[1]	InvMixColumns
SubBytes	InvShiftRows
ShiftRows	InvSubBytes
AddRoundKey[2]	AddRoundKey[0]

Same Algorithm for both Enc. and Dec.

2 Round Encryption	**2 Round Decryption**
AddRoundKey[0]	AddRoundKey[2]
SubBytes	InvSubBytes
ShiftRows	InvShiftRows
MixColumns	AddRoundKey[1]
AddRoundKey[1]	InvMixColumns
SubBytes	InvSubBytes
ShiftRows	InvShiftRows
AddRoundKey[2]	AddRoundKey[0]

Commutative

Same Algorithm for both Enc. and Dec.

- Notice,

  InvMixCol(State $\oplus$ Key) = InvMixCol(State) $\oplus$ InvMixCol(Key)

- This is possible because both XOR operation and InvMixCol operation are linear
- Also, w.r.t XOR operation

$$a \oplus b = b \oplus a$$

- Therefore, if we denote InvMixCol(Key) by AddRoundKey', then …

Same Algorithm for both Enc. and Dec.

2 Round Encryption

AddRoundKey[0]

SubBytes
ShiftRows
MixColumns
AddRoundKey[1]

SubBytes
ShiftRows
AddRoundKey[2]

2 Round Decryption

AddRoundKey[2]

InvSubBytes
InvShiftRows
AddRoundKey[1]
InvMixColumns

Operations can be interchanged

InvSubBytes
InvShiftRows
AddRoundKey[0]

Same Algorithm for both Enc. and Dec.

2 Round Encryption	**2 Round Decryption**
AddRoundKey[0]	AddRoundKey[2]
SubBytes	InvSubBytes
ShiftRows	InvShiftRows
MixColumns	InvMixColumns
AddRoundKey[1]	AddRoundKey’[1]
SubBytes	InvSubBytes
ShiftRows	InvShiftRows
AddRoundKey[2]	AddRoundKey[0]

Same Algorithm for both Enc. and Dec.

- Now the sequence of operations in encryption as well as decryption are same
  - Same circuitry can be used to do enc. as well as dec.

- To achieve the above, *MixColumns* operations was omitted in the last round
  - HOMEWORK: Try to think why !!!

AES Key Schedule Algorithm

- Some Notations:

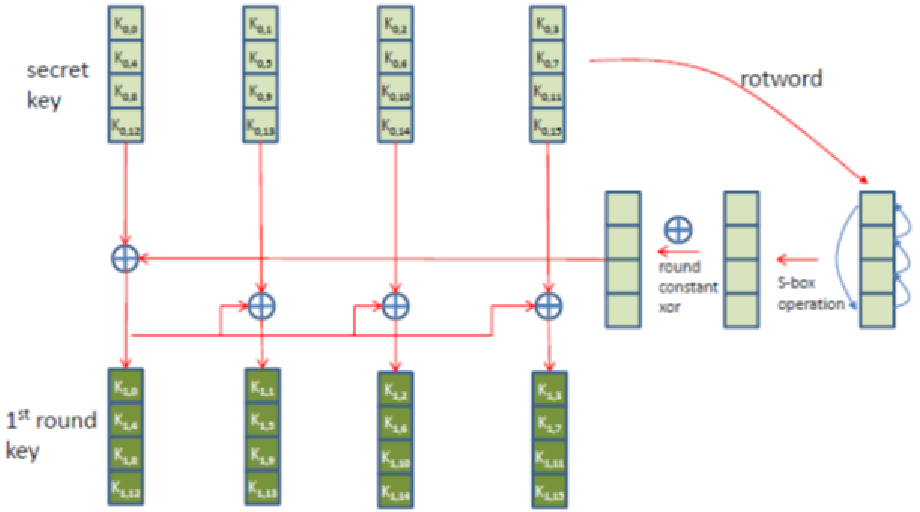
1. Word : One word = 32-bits (4 bytes)

   Each round key and intermediate state has 4 words (128-bits)

   As AES-128 has 11 round keys, it requires 44 words

AES-128 Key Schedule Algorithm

```
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 mod 4 = 0)    temp = SubWord (RotWord (temp))
                                ⊕ Rcon[i/4];
     w[i] = w[i−4] ⊕ temp
    }
}
```



AES Key Schedule Algorithm (KSA)

- RotWord(a,b,c,d) = (b,c,d,a)
- SubWord(p,q,r,s) = (S(p),S(q),S(r),S(s))
- Rcon[i] = (RC[i],0,0,0) where, RC[1] = 1, RC[i] = 2 $\times$ RC[i-1] (multiplication in GF(2^8))

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

- AES-192/256 KSA also work in a similar fashion
  - For more details, please refer to the official AES document: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Success of AES

- None of the attacks break AES much better than brute force
  - Except side channel attack (SCA)
  - However, it is applicable to specific scenario only
- Intel chips have special instruction sets for AES. Allows extremely efficient implementation. Can resist many SCA as well.
- Every famous Cryptographic library used on the web has an implementation of AES in it.
- “No one ever lost his job for using AES” ☺

AES NI

- Advanced Encryption Standard New Instructions
- Accelerating AES on modern Intel and AMD processors with dedicated instructions
- Using the AES NI instruction set, the program can do a whole round in a single instruction

Instruction	Description[2]
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply (CLMUL).[3]