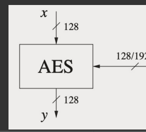
Advanced Encryption Standard

→ Ex: Web security TLS, IPsec, IEEE 802.11;

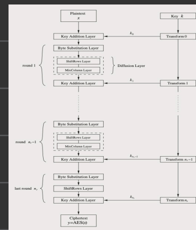


128 bits ⇒ $n_r = 10$
192 ⇒ $n_r = 12$
256 ⇒ $n_r = 14$
that is why n_r is low

→ AES cipher is almost identical to Rijndael Block Cipher (block & key size varies between 128, 192 & 256 bits) → AES only needs block size of 128 bits

→ AES is different from DES since it has no fiestel network (which works on a subset of a block's text in one iteration) → AES encrypts all 128 bits in 1 iteration

↳ Each AES round is a layer manipulating 128 bits until Mix Column Transformation x = plain, y = cipher, n_r = # of rounds



→ 3 layers:

1. Key Layer: 128-bit round key derived from main key XORed to state
2. Byte Subst. Layer (S-box): Each element of state nonlinearly transformed using lookup tables.
3. Diffusion Layer: Ensures changes of individual bit propagate across 128 bits
   ↳ ShiftRows Sublayer: permutes data on byte layer
   ↳ MixColumn Sublayer: Matrix Operation combining blocks of 4 bytes

confusion: changes in individual bits ↗ ⇒ non linear changes in state

key schedule computes $(k_0, \ldots, k_{n_r})$ → Use Galois Fields

→ Galois / Finite Field: Set w/ finite number of elements of which we can add, subtract, multiply & invert

→ Def^n of Groups: Set of elements G together w/ operation ∘ that combines 2 elements of G

↳ ∘ is closed: $\forall a, b \in G \; (a \circ b \in G)$

↳ ∘ is associative, commutative

↳ $\exists\, 1 \in G$ called Neutral / Identity Element → $\forall a \in G \; (a \circ 1 = 1 \circ a = a)$

↳ $\forall a \in G \; \exists a^{-1} \in G \; (a \circ a^{-1} = a^{-1} \circ a = 1)$ → Inverse of a

Ex: $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ w/ addition & modulo m form group w/ neutral element 0

$a + (-a) \equiv 0 \mod m$ → no inverse ∴ no group

NEED additive & multiplicative abelian group → Fields

→ Def^n of Field: Set of elements F w/ following → Order = # of elements

↳ all $f_i \in F$ form additive abelian group w/ operation + and neutral 0

↳ all $f_i \in F - \{0\}$ form multiplicative abelian group w/ op. × and neutral 1

↳ When 2 group ops mixed, distributivity: $a(b+c) = ab + ac \quad \forall a, b, c \in F$

Ex: $\mathbb{R}$ is a field w/ neutral = 0 for additive group & 1 for multiplicative group.

Additive inverse $-a$

Multiplicative: $1/a$ for $a \neq 0$

$GF(p^m) \equiv$ Field of p^m elements; q: characteristic

→ Theorem: A field w/ order q only exists if q is a prime power $q = p^m$ for some prime p & $m \in \mathbb{Z}^+$

→ Ex: Fields of 11 elements, 81 elements or 256 elements.

→ Theorem: Let p be a prime. $\mathbb{Z}_p = GF(p)$ is a Prime / Galois Field w/ a prime number of elements $\forall a \in GF(p) \mid a \neq 0$ • a has an inverse

+ and • operations are done via modulo p

Ex: Smallest prime field $GF(2) = \mathbb{Z}_2$ → AES uses 256 elem fields $GF(2^8)$ → each field element represented by 1 byte for manipulation → 2^8 not prime so + & • cannot use modulo

addition

+	0	1
0	0	1
1	1	0

multiplication

×	0	1
0	0	0
1	0	1

→ Extension Field: Fields $GF(m)$ for $m > 1$ where elements represented as polynomials & polynomial arithmetic → max degree of $m-1$

→ Ex: $\forall A \in GF(2^8)$: $A(x) = a_7 x^7 + \cdots + a x + a_0$ for $a_i \in GF(2)$

$$A = \begin{bmatrix} a_7 \\ a_6 \\ \vdots \\ a_0 \end{bmatrix}$$

Irreducible polynomials needed

→ Addition & Subtractions in Extension Fields: For $A(x), B(x) \in GF(2^m)$ ⇒ $C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i$ where $c_i = a_i + b_i \mod 2$

⇒ $C(x) = A(x) - B(x) = \sum_{i=0}^{m-1} c_i x^i$ where $c_i = a_i - b_i \mod 2$

→ Multiplication in Extension Fields: $A(x) B(x) = (a_{m-1} x^{m-1} + \cdots + a_0)(b_{m-1} x^{m-1} + \cdots + b_0) = C'(x) = c'_{2m-2} x^{2m-2} + \cdots + c'_0$

$c'_0 = a_0 b_0 \mod 2$
$c'_1 = a_0 b_1 + a_1 b_0 \mod 2$
$c'_{2m-2} = a_{m-1} b_{m-1} \mod 2$

→ Idea: Product of multiplication divided by certain polynomial & only consider only remainder after polynomial division

→ Def^n of EF Multiplication: $\forall A(x), B(x) \in GF(2^m)$. $P(x) \equiv \sum_{i=0}^{m} p_i x^i$ for $p_i \in GF(2)$ is an irreducible polynomial of degree m. $C(x) = A(x) \cdot B(x) \mod P(x)$

→ Inversion in $GF(2^8)$ is used heavily in the Byte Subst. Layer containing S-boxes → Given $GF(2^m)$ & irreducible reduction polynomial $P(x)$, $\forall A \in GF(2^m)$: $A^{-1}(x) \cdot A(x) \equiv 1 \mod P(x)$

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

$P(x) = x^8 + x^4 + x^3 + x + 1$
$GF(2^8)$
Bytes xy within AES S-box

Ex: $x^7 + x^6 + x = (1100\,010)_2 = (C2)_{hex} = (xy)$

Row C, Column 2

$(2F)_{hex} = (0010\ 1111)_2 = x^5 + x^3 + x^2 + x + 1$

$(x^7 + x^6 + x)(x^5 + x^3 + x^2 + x + 1) \equiv 1 \mod P(x)$

128 bit ↑ 192 bit ↑ 256 bit ↑
subkey k_i also arranged in 4×4, 4×6 or 4×8 matrix

→ State A is the following

A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

$(A_0, \ldots, A_{15})$ fed byte-wise into S-box → $(B_0, \ldots, B_{15})$

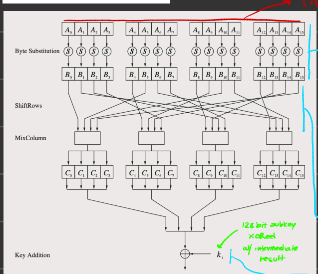
Permuted byte-wise in ShiftRows layer & mixed by MixColumn Transf. $c(x)$

ByteSub(A) + ByteSub(B) ≠ ByteSub(A + B)

Ex: $S((C2)_{hex}) = (25)_{hex}$

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

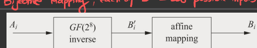
NO SUCH A_i s.t. $S(A_i) = A_i$



→ Byte Subst. Layer: 16 parallel S-boxes, each of I/O size = byte (identical) $S(A_i) = B_i$ → only nonlinear component

Bijective Mapping, each of $2^8 = 256$ possible inputs mapped one-to-one output

Decryption: reverse S-box



$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \equiv \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \mod 2$$

After 3 rounds, every byte of state depends on all 16 plaintext bytes.

$B'_i = A_i^{-1}$ $P(x) = x^8 + x^4 + x^3 + x + 1$ $GF(2^8)$

Destroys algebraic structure of Galois Field → prevents attacks that leverage inversion.

→ Diffusion Layer: ShiftRows + MixColumn Transformations

DIFF(A) + DIFF(B) = DIFF(A + B) → Linear Transformation mixing each column of state matrix   MixColumn(B) = C

Each byte column is vector & multiplied by 4×4 matrix → First 4 bytes

Ex: $B_0 = x^3 + x^2$ $B_5 = x^7 + 1$ $B_{10} = x^5 + x^4 + 1$ $B_{15} = x^5 + x^4 + x^3$

$C_0 = 02\,B_0 + \ldots$??

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

→

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	⟶ three positions right shift
B_{10}	B_{14}	B_2	B_6	⟶ two positions right shift
B_{15}	B_3	B_7	B_{11}	⟶ one position right shift

cyclically shifts 2^{nd} row of state matrix 3 bytes to right, 3^{rd} row 2 bytes to right & 4^{th} row 1 byte to right.

→ Key Addition Layer: 16 byte state matrix ⊕ 128 bits subkey

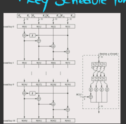
number of subkeys = # of rounds + 1

→ Key Schedule: Takes original input key & derive subkeys → key whitening: XOR addition in the beginning & end of AES → AES subkeys computed recursively

↳ Word oriented → each word = 32 bits where subkeys stored in key expansion array W

→ Key Schedule for 128-bit AES: Each 128-bit subkey consists of 4 words

11 subkeys stored in W w/ 11×4 = 44 elements $W[0], \ldots, W[43]$ $K_0, \ldots, K_{15}$ = Bytes of AES key



$RC[1] = x^0 = (0000\ 0001)_2$
$RC[2] = x^1 = (0000\ 0010)_2$
⋮
$RC[10] = x^9 = (0011\ 0110)_2$

g adds nonlinearity to key schedule & removes symmetry in AES

$W[4i] = W[4(i-1)] \oplus g(W[4i-1])$ for $i \in \mathbb{Z}_{11}$ and g is a nonlinear f^n w/ 4-byte I/O

↓

$W[4i + j] = W[4i + j - 1] \oplus W[4(i-1) + j]$ $i \in \mathbb{Z}_{11}$, $j = 1, 2, 3$

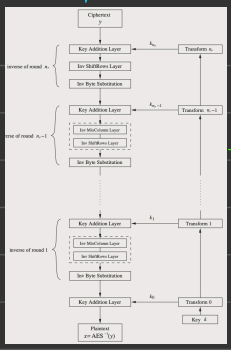
g rotates 4 input bytes, performs byte wise S-box Subst. & adds round coeff RC to it. → $RC \in GF(2^8)$

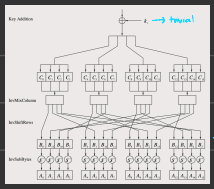
→ In general, when implementing any of 3 key schedules, 2 approaches.

↳ Precomputation: All subkeys expanded into W. Encrypt/Decrypt executed afterward → PC/server implementations where largest pieces of data are encrypted under one key

↳ On-the-fly: New subkey derived for next AES round during encrypt/decrypt where decryption requires recursive derivations of all subkeys → overhead, slower than encrypt

→ For decryption, all layers must be inverted (each being similar to original) where order of subkeys is reversed





→ After addition of subkey, Inv. MixColumn applied to state → inverse of matrix used in MixColumn must be calculated

→ To do Inv ShiftRows inverse by shifting same rows left.

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

→

B_0	B_4	B_8	B_{12}	no shift
B_{13}	B_1	B_5	B_9	← three positions left shift
B_{10}	B_{14}	B_2	B_6	← two positions left shift
B_7	B_{11}	B_{15}	B_3	← one position left shift

→ To do Inv S-box → do: $A_i = S^{-1}(B_i) = S^{-1}(S(A_i))$ → compute inverse of affine transformation on each $B_i = (b_7, \ldots, b_0) = B_i(x)$ and $(b_7', \ldots, b_0')$

$A_i = (B_i')^{-1} \in GF(2^8)$

$= (a_7, \ldots, a_0)$

everything done in $GF(2^8)$

additions in vector-based multiplication is XOR

⇒ Decryption round 1 need last subkey, ...

⇒ need $n_r + 1$ subkeys in reversed orders