

Rijndael: The Advanced Encryption Standard

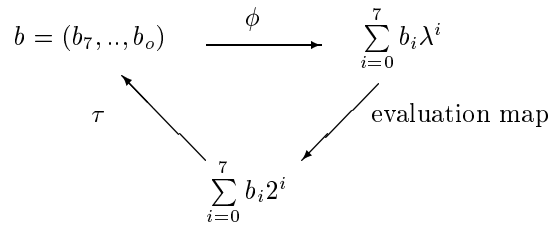
as recapped by R.E. Hartwig

The Advanced Encryption Standard (AES for short) is a symmetric block cipher in the classical sense. It can operate in several modes such as the ECB, CBC and CFB modes. Recall that these acronyms stand for “electronic code book”, “cipher block chaining” and “cipher feedback”.

AES admits key lengths using 128, 192 and 256 bits. We shall for convenience only examine the former.

Overall Picture

- AES maps 128 bit blocks into 128 bit blocks.
- It is made up of 10 rounds. Each round has a 128 bit key k_i $i = 1, 2, \dots, 10$ derived from one original key k_o . As such there is a *key generation schedule*
- There is a round zero, which uses the original key k_o .
- Each round has 4 layers or steps. These really are just 4 specially cooked up functions that send 128 bit blocks into 128 bit blocks.
- Rather than working with 128 bit blocks, the mapping work with 4×4 matrices, whose entries are bytes. Recall that *all* computer transmission is in term of bytes. A byte is an 8 bit string or row-vector $b = (b_7, \dots, b_1, b_0) = \mathbf{b}^T \in \mathbb{Z}_2^8$. We can associate with each such byte b three mappings:



(i) The map $\tau : \{0, 1, \dots, 255\} \rightarrow \mathbb{Z}_2^8$ defined by $\tau(a) = \tau(\sum_{i=0}^7 a_i 2^i) = (a_7, \dots, a_0)$ is the well known *binary expansion* map. It is one-to-one and onto.

(ii) the polynomial map $\phi : \mathbb{Z}_2^8 \rightarrow \mathbb{Z}_2[\lambda]$ defined by $\phi(b) = \sum_{i=0}^7 b_i \lambda^i \in \mathbb{Z}_2[\lambda]$ is also one-to-one and onto. We shall often use “x” instead of λ in our polynomials.

It will be to our advantage to think of a byte as a string, a real number or as polynomial, depending on the setting. Whenever we need “wrap-around”, we shall use the latter.

We divide a given a 128 bit block M into 16 bytes $(M_o, M_1, \dots, M_{15})$ and store these into a 4×4 matrix

$$A = \begin{bmatrix} M_o & M_4 & \dots & M_{12} \\ M_1 & M_5 & \dots & M_{13} \\ M_2 & \dots & \dots & M_{14} \\ M_3 & \dots & \dots & M_{15} \end{bmatrix} = [a_{ij}].$$

Now in order to manipulate these “byte matrices” from $M_4(\mathbb{Z}_2[\lambda])$,

we must know how to operate with two bytes. This we now pursue.

Addition of two bytes poses no problem. It is done element-wise:

$$(a_7, \dots, a_0) + (b_7, \dots, b_0) = (a_7 + b_7, \dots, a_0 + b_0),$$

where $a_i + b_i$ is performed in \mathbb{Z}_2 .

To multiply two polynomials $(\sum_{i=0}^7 a_i \lambda^i) * (\sum_{i=0}^7 b_i \lambda^i)$ in the normal way, produces a new polynomial $\sum_{i=0}^{14} c_i \lambda^i$, which **no longer** corresponds to a byte! To avoid “overflow” we need to introduce periodicity or “wrap-around”, which ensures “closure”, in the sense that the new polynomial $c(\lambda)$ will again be of degree 7 and again corresponds to a unique byte.

To do this we follow the same steps as were taken to create \mathbb{Z}_p^* . We start by taking a prime polynomial $\pi(\lambda)$ with $\partial(\pi) = 8$ and consider the *remainders* of all polynomials over \mathbb{Z}_2 modulo π . This construction will yield a (the) **finite field** $GF(2^8)$ with 256 elements. This is one of **the** most important constructions in algebra.

Consider $\pi(x) = x^8 + x^4 + x^3 + x + 1$, and define $p(x) \sim q(x)$ iff $\pi|(p - q)$. (also written as $p \equiv q$). It is easily

seen that this is an equivalence (RST) relation, with equivalence classes $\overline{p(x)} = \{q(x); q \sim p\}$. In particular, if $p(x) = q(x)\pi(x) + r(x)$, with $r(x) = 0(x)$ or $\partial(r) < 8$, the $\bar{p} = \bar{r}$. Again $r(x)$ is the unique representative of \bar{p} of degree less than 8. Needless to say this parallels the idea of a CSRs (Complete Systems of Residues).

We now consider $\mathbb{F} = \{r_0 + r_1x + \dots + r_7x^7; r_i \in \mathbb{Z}_2, \text{ mod } \pi(x)\} = \mathbb{Z}_2[x]/\pi(x)$. We note that

1. $\#(\mathbb{F}) = 2^8$
2. \mathbb{F} is closed under addition and (polynomial) multiplication. In particular $\overline{pq} = \bar{r}$, where r is the remainder of pq after division by π .
3. every non-zero element has an inverse! Indeed, if $a \in \mathbb{F}$, then $(a(x), \pi(x)) = 1$ and hence by Euclid, $a(x)s(x) + \pi(x)t(x) = 1$ for some $s(x), t(x)$. This means that $\bar{a}\bar{s} = \bar{1}$ and $\bar{s} = \bar{a}^{-1}$ in \mathbb{F} . In other words, $s(x)$ is the inverse of $a(x)$ **modulo** π . The remaining field axioms are easily checked, and because there are only two choices for each r_i , it is clear that \mathbb{F} has 256 elements.

When we multiply two byte matrices we think of the bytes as polynomials over \mathbb{Z}_2 and perform the multiplication mod π i.e. $A \in M_4[\mathbb{Z}_2[\lambda]/\pi]$. That is, rather than using 128 bit strings we use 4×4 matrices over $GF(2^8)$.

Example 0.1. $(1, 0, 0, 1, 1, 0, 1, 1) = x^7 + x^4 + x^3 + x + 1$ and $(0, 0, 0, 0, 0, 0, 0, 1) = 1(x)$. which is the multiplicative identity.

There exist **four** special maps that are applied in succession to a 4×4 input matrix A. There are:

- (i) α is a “byte substitution” (BS) map
- (ii) β is a “row-shift” (RS) map
- (iii) γ is a Column-Mix (CM) map
- (iv) δ is a “Round Key Addition (RKA) map.

Thus

$$A \rightarrow B = \alpha(A) \rightarrow C = \beta(\alpha(A)) \rightarrow D = \gamma(\beta(\alpha(A))) \rightarrow E = \delta(\gamma(\beta(\alpha(A))))$$

In which both the input A and the output E are 4×4 .

- (a) The map α is needed against differential and linear crypto attacks.
- (b) the map β is used to diffuse bits over multiple rounds.
- (c) the map δ uses the key K_i during the i-th round.

The four maps

The map α

The map α is defined entry wise by $A \rightarrow \alpha(A) = [\alpha(a_{ij})]$. We are given a **fixed** 16×16 matrix S (called an S-box). Its entries come from $\{0, 1, \dots, 255\}$ and we shall see shortly how it is constructed.

If a_{ij} is the byte (a,b,c,d,d,e,f,g,h), then

$$\alpha(a_{ij}) = \tau[S_{\tau^{-1}(a,b,c,d), \tau^{-1}(e,f,g,h)}],$$

where $\tau^{-1}(a, b, c, d)$ and $\tau(e, f, g, h)$ are numbers in $\{0, 1, \dots, 15\}$. These give a position in the matrix S. The bin-ex of the entry in this position gives the output byte.

Example 0.2. $(1, 0, 0, 0, 1, 0, 1, 1) = ((1, 0, 0, 0), (1, 0, 1, 1)) = (8, 11)$. These give $S_{8,11}$, which we look up as 61. Expanding this gives the byte $(0, 0, 1, 1, 1, 1, 0, 1)$.

This may be inverted as:

$(0, 0, 1, 1, 1, 1, 0, 1) \rightarrow 2^5 + 2^4 + 2^3 + 2^2 + 1 = 61$. This we look up in S, and find it in the (8,11) position. We binary-expand the coordinates 8 as (1,0,0,0) and 11 as (1,0,1,1), to give the byte (1,0,0,0,1,0,1,1).

The map β

Suppose that $B = \begin{bmatrix} \mathbf{b}_0^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{b}_3^T \end{bmatrix}$ is partitioned onto *rows* and that $\Omega = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. The map β is defined by

$$C = \beta(B) = \begin{bmatrix} \mathbf{b}_o^T \\ \mathbf{b}_1^T \Omega \\ \mathbf{b}_2^T \Omega^2 \\ \mathbf{b}_3^T \Omega^3 \end{bmatrix}$$

The effect of this is to rotate the rows of B to the left, 0,1,2,3 steps respectively.

The map γ

The column mix map is defined as the matrix multiplication:

$$D = \gamma(B) = TC,$$

where T has three equivalent form:

$$T = \begin{bmatrix} (00000010) & (00000011) & (00000001) & (00000001) \\ (00000001) & (00000010) & (00000011) & (00000001) \\ (00000001) & (00000001) & (00000010) & (00000011) \\ (00000011) & (00000001) & (00000001) & (00000010) \end{bmatrix},$$

$$\phi(T) = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & 1+x & 1 \\ 1 & 1 & x & 1+x \\ 1+x & 1 & 1 & x \end{bmatrix}, \text{ and } \tau^{-1}(T) = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

Again the multiplication is done using polynomials mod π i.e. $D = \phi^{-1}[\phi(T)\phi(C)]$.

It should be noted that

- (i) T is a **circulant** matrix of the form $2I + 1.\Omega + 1.\Omega^2 + 3\Omega^3 = xI + 1.\Omega + 1.\Omega^2 + (x+1)\Omega^3$,
- (ii) $\det[T(x)] = 1$, as a λ -matrix over \mathbb{Z}_2 .
- (iii) T is invertible with an inverse that is again a λ -matrix. This matrix is determined by its first row or column. Indeed

$$T^{-1} = (1110)I + (1001)\Omega + (1101)\Omega^2 + (1011)\Omega^3,$$

$$\phi(T)^{-1} = (x^3 + x^2 + x)I + (x^3 + 1)\Omega + (x^3 + x^2 + 1)\Omega^2 + (x^3 + x + 1)\Omega^3$$

$$(\tau^{-1}(T))^{-1} = 14I + 11\Omega + 13\Omega^2 + 9\Omega^3 = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}$$

The map δ

During round i ($i = 1,2,\dots,10$), the ‘‘round key addition’’ map, adds (i.e. XORs) the i -th key matrix K_i to the output of the previous layer. That is

$$\delta_i : D \rightarrow E = \delta_i(D) = D \oplus K_i.$$

We note in passing that clearly $\delta^{-1} = \delta$ and that T is invertible.

Key Generation

Suppose that K_o is the 128 bit key which has been inscribed column wise into the 4×4 matrix $K_o = [\mathbf{w}_o, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3]$. We shall use the 4 columns \mathbf{w}_i , $i = 0, \dots, 3$, to generate 44 columns: $\mathbf{w}_o, \dots, \mathbf{w}_{43}$, which in turn give the 11 key matrices:

$$K_o = [\mathbf{w}_o, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3], K_1 = [\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6, \mathbf{w}_7], \dots, K_{10} = [\mathbf{w}_{40}, \mathbf{w}_{41}, \mathbf{w}_{42}, \mathbf{w}_{43}].$$

The additional columns \mathbf{w}_i are formed as follows.

Case(a) If $i \neq 4k$, then $\mathbf{w}_i = \mathbf{w}_{i-4} + \mathbf{w}_{i-1}$ (\mathbb{Z}_2 addition)

Case(b) If $i = 4k$, then $\mathbf{w}_i = \mathbf{w}_{i-4} + \chi(\mathbf{w}_{i-1})$.

If $\mathbf{w}_{i-1} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$, then χ is made up of the following string of operations:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \xrightarrow{\Omega} \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \xrightarrow{\alpha} \begin{bmatrix} \alpha(b) \\ \alpha(c) \\ \alpha(d) \\ \alpha(a) \end{bmatrix} \rightarrow \begin{bmatrix} \alpha(b) + r(i) \\ \alpha(c) \\ \alpha(d) \\ \alpha(a) \end{bmatrix}.$$

In this $r(i)$ is the ‘‘round number’’, which is given by $r(i) = (00000010)^{(i-4)/4}$ ($i = 4k$), and which is computed from $\mathbf{x}^k \bmod \pi$, where $k = i/4$. This is done for $i = 4, 5, \dots, 43$, giving the 10, new key matrices $K_j = [\mathbf{w}_{4j}, \mathbf{w}_{4j+1}, \mathbf{w}_{4j+2}, \mathbf{w}_{4j+3}]$.

The S-box

We start with the 16×16 empty matrix and only consider the position (p, q) in the array, with $0 \leq p, q < 16$. Let $\tau(p) = (p_3, \dots, p_o)$ and $\tau(q) = (q_3, \dots, q_o)$. As such (p, q) induces a byte $(p_3, \dots, p_o, q_3, \dots, q_o) = (g_7, \dots, g_o) = g$. This in turn gives the polynomial $\phi(g) = \sum_{i=0}^7 g_i x^i$. We then compute $h(x) = g(x)^{-1} \bmod \pi$ as an element in $\text{GF}(256)$, which gives us the string $(h_7, \dots, h_o) = \mathbf{h}^T$. We now define the 8×8 circulant matrix $L = I_8 + \Omega + \Omega^2 + \Omega^3 + \Omega^4$, and form the output vector:

$$\mathbf{z} = L(F\mathbf{h}) + \mathbf{u}_o$$

where $F\mathbf{h} = \begin{bmatrix} h_o \\ \vdots \\ h_7 \end{bmatrix}$ and $\mathbf{u}_o = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

From the column vector \mathbf{z} , we obtain the output byte (z_7, \dots, z_o) and the associated real value $\sum_{i=0}^7 z_i 2^i$ in $[0, 256)$, which becomes $(S)_{pq}$.

The complete Rijndael S-box is given by:

	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
0	99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
1	202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
2	183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
3	4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
4	9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
5	83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
6	208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
7	81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
8	205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
9	96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
10	224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
11	231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
12	186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
13	112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
14	225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
15	140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Example 0.3. Let us compute $s_{12,11}$. First we compute $\tau(12) = (1100)$ and $\tau(11) = (1011)$, and then form $g = (11001011)$, which corresponds to $g(x) = x^7 + x^6 + x^3 + x + 1$. To compute $g^{-1} \bmod \pi$, we use Euclid to give us $(x^7 + x^6 + x^3 + x + 1)(x^2) + (x^8 + x^4 + x^3 + 1)(x + 1) = 1$. This shows that $h(x) = x^2$. We then form

$$\mathbf{z} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Reversing the order we obtain the output byte (00011111) which sums to the number 31. Hence we set $(S)_{12,11} = 31$.

The inversion map $g \rightarrow g^{-1} \pmod{\pi}$ is highly non-linear. This is combined with matrix multiplication and addition to avoid some of the more serious attacks. Note that L is again a circulant of a particularly simple form. The vector \mathbf{u}_o guarantees that *no* input vector will ever equal its S-box output or that of its complement (add \mathbf{e}).

Description

It is easily seen that each of the 4 maps is invertible.

1. α^{-1} acts as follows:
 $((r, s, t, u)(v, w, x, y)) \rightarrow \tau^{-1}(r, s, t, u, v, w, x, y)$ which is a real number t in $[0,256)$. We look up this number t in the S matrix, and find that, say, $t = (S)_{p,q}$. We then form $(\tau(p), \tau(q))$ which gives us the output byte.
2. It is trivial that β^{-1} shifts the rows of the input matrix to the right. That is,
$$\begin{bmatrix} \mathbf{a}_o^t \\ \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{a}_o^T \\ \mathbf{a}_1^T \Omega^{-1} \\ \mathbf{a}_2^T \Omega^{-2} \\ \mathbf{a}_3^T \Omega^{-3} \end{bmatrix}.$$
3. The inverse of γ is given by $C = T^{-1}D$ or rather $C(\lambda) = T(\lambda)^{-1}D(\lambda)$, where $\mathbb{T}(\lambda)^{-1}$ is given as before.
4. Since addition over \mathbb{Z}_2 is involutory, it immediately follows that $\delta^{-1} = \delta$.

Decipherment

Before we address the decipherment question we first observe that the 4 maps satisfy

Lemma 0.1. (i) $\alpha\beta = \beta\alpha$

(ii) If f_{TK} is the affine map defined by $f_{TK}(X) = TX + K$, then its inverse is given by $f_{TK}^{-1} = f_{T^{-1}, T^{-1}K}$

Proof: (i) Since α acts element-wise, shifting rows commutes with this operation. This also ensures that $\alpha^{-1}\beta^{-1} = \beta^{-1}\alpha^{-1}$.

(ii) $f_{T^{-1}, T^{-1}K}(TX + K) = T^{-1}(TX + K) + T^{-1}K = X$. Because $f_{TK} = \delta\gamma$, this result implies that $\gamma^{-1}\delta^{-1} = (\delta\gamma)^{-1} = f_{T^{-1}, T^{-1}K} = \delta'\gamma^{-1}$, where $\delta'(X) = X + T^{-1}K$. In the i -th round this means that $(\delta_i\gamma)^{-1} = \delta'_i\gamma^{-1}$, where $\delta'_i(X) = X + T^{-1}K_i$.

Consider now the encipherment rounds:

$$\begin{array}{ll} \text{round 0 :} & \delta_o \\ \text{round 1 :} & \alpha \quad \beta \quad \gamma \quad \delta_1 \\ \text{round 2 :} & \alpha \quad \beta \quad \gamma \quad \delta_2 \\ & \vdots \\ \text{round 9 :} & \alpha \quad \beta \quad \gamma \quad \delta_9 \\ \text{round 10 :} & \alpha \quad \beta \quad \delta_{10} \end{array}$$

When we *decipher*, we invert each map in the sequence and reverse the order. In addition we now use the facts

that $\alpha^{-1}\beta^{-1} = \beta^{-1}\alpha^{-1}$ and that $(\delta_i\gamma)^{-1} = \delta'_i\gamma^{-1}$ and regroup the the rounds as follows:

invert:

$$\begin{array}{cccc}
 \delta_{10}^{-1} & & & \\
 \beta^{-1} & \alpha^{-1} & \delta_9^{-1} & \gamma^{-1} \\
 \beta^{-1} & \alpha^{-1} & \delta_8^{-1} & \gamma^{-1} \\
 \vdots & & & \\
 \beta^{-1} & \alpha^{-1} & \delta_1^{-1} & \gamma^{-1} \\
 \beta^{-1} & \alpha^{-1} & \delta_o^{-1} & \\
 \end{array}
 \longrightarrow \text{Regroup:}
 \begin{array}{cccc}
 \delta_{10} & & & \\
 \alpha^{-1} & \beta^{-1} & \gamma^{-1} & \delta'_9 \\
 \alpha^{-1} & \beta^{-1} & \gamma^{-1} & \delta'_8 \\
 \vdots & & & \\
 \alpha^{-1} & \beta^{-1} & \gamma^{-1} & \delta'_1 \\
 \alpha^{-1} & \beta^{-1} & \delta_o^{-1} & \\
 \end{array}
 .$$

This tells us that to decrypt we

1. apply δ_o using the 10-th round key K_{10}
2. execute 9 rounds with $\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta'_{10-i}$, using the keys K_9, \dots, K_1 respectively.
3. for the last round apply α^{-1}, β^{-1} and δ_o using K_o .

This means that we essentially have the *same* structure as for encipherment. The extra γ in the last round would have messed up the parallel between the encipherment and decipherment algorithms.

For 8 bit processors decryption is slower than encryption. Also γ^{-1} i.e. T^{-1} , is more complicated than T , which makes decryption 30% slower! In many cases decryption is not needed as is the case with cipher feedback. Because en- and de-cryption are not entirely the same there are no “weak” keys as exists for the DES system.

Remarks

1. The Rijndael system differs from the Feistel system. In the latter 1/2 of the bits are moved but not changed in each round.
2. In the AES *all* bits are changed each time, which means faster diffusion.
3. two rounds suffice for complete diffusion, after which each output bit depends on all the 128 input bits.
4. there is no trapdoor in the S boxes.
5. the use of the inverse map $g \rightarrow g^{-1}$ is highly non-linear and resists numerous sophisticated types of attack, such as linear and differential attacks.
6. The row-shift map β was added to spoil the “square” and “truncated differential “ attacks.
7. The column mix map γ ensures diffusion. The change of just 1 input byte, changes all 4 output byte.
8. the key schedule uses non-linear mixing of the key bits via the S boxes. This is aimed at spoiling attacks where part of a key may be known.
9. 7 rounds suffice to prevent a brute force attack.
10. Byte multiplication is done mod (100011011).

If $b_7 = 0$, then $\lambda b = (b_6, b_5, b_4, b_3, b_2, b_1, b_o, 0)$.

If $b_7 = 1$ then λb is obtained by shifting b , adding π and dropping the overflow i.e. the first zero bit. Thus

$$\begin{array}{cccccccc}
 (b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_o & 0) \\
 (1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1) + . \\
 \hline
 (0 & b_6 & b_5 & b_4 & 1 + b_3 & 1 + b_2 & b_1 & b_o & 1)
 \end{array}$$

Hence $\lambda b = (b_6, b_5, b_4, 1 + b_3, 1 + b_2, b_1, 1 + b_o, 1)$ and

$$(1 + \lambda)b = (b_7 + b_6, b_6 + b_5, b_5 + b_4, b_4 + b_3 + 1, b_3 + b_2 + 1, b_2 + b_1, b_1 + b_o + 1, b_o + 1)$$

Exercise 0.1. (a) Show that $\det[\Gamma(x)] = 1$

(b) Show that $T(x)J = J$, where $J = ee^T$

(c) If $M = 2^{127} + 1 = (1, 0, \dots, 0, 1)$ is a 128 bit message input string, and $K_0 = 1 = (0, \dots, 0, 1)$ is the 128 bit key string, compute the 128-bit output string after round zero and one round of AES.

(d)