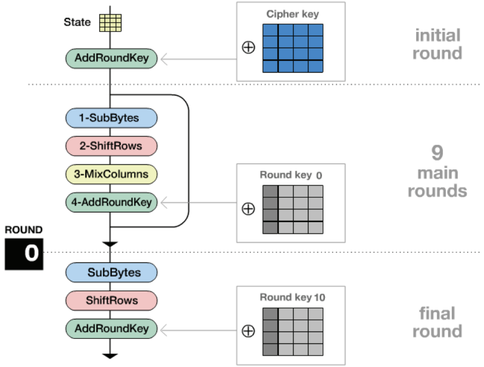
Advanced Encryption Standard



Finite Field Operations in Rijndael & AES

- The Rijndael data block size can be 128, 192 or 256 bits; however, the AES is fixed at 128 bits only
- The state matrix is formed from the input data as a 4×4, 4×6, and 4×8 matrices, for 128, 192 or 256 bits, respectively
- Given the 128-bit data $(A_0 A_1 A_2 \cdots A_{14} A_{15})$ such that each of A_i is 8 bits (1 byte), the 4×4 state matrix is formed as

$$\begin{bmatrix} A_0 & A_4 & A_8 & A_{12} \\ A_1 & A_5 & A_9 & A_{13} \\ A_2 & A_6 & A_{10} & A_{14} \\ A_3 & A_7 & A_{11} & A_{15} \end{bmatrix}$$

The 8-bit (1-byte) binary data is usually represented in hexadecimal, such as $(\text{a3}) = (1010\ 0011)$

Finite Field Operations in Rijndael & AES

- While the 8-bit input data block is a binary number in its most generic form, the Rijndael/AES treats each one of the bytes in the state matrix as elements of the Galois field GF(2^8)
- The irreducible polynomial of the field GF(2^8) is $p(x) = x^8 + x^4 + x^3 + x + 1$
- A field element $a(x) \in$ GF(2^8) is represented using a polynomial of degree at most 7 with coefficients $a_i \in$ GF(2) such that

$$\sum_{i=0}^{7} a_i x^i = a_7 x^7 + x_6 x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

For example, (a3) = (1010 0011) = $x^7 + x^5 + x + 1$

Field Addition in GF(2^8)

- Given two field elements $a(x)$ and $b(x)$, their sum is $c(x)$ is computed by polynomial addition

$$c(x) = a(x) + b(x)$$

  such that $c_i = a_i + b_i \pmod 2$ for $0 \leq i \leq 7$, as shown below:

$$\begin{array}{ccccccccc} & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ \oplus & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ \hline & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \end{array}$$

  where $\oplus$ is the XOR operation, which is the same as modulo 2 addition

- As the degree of the sum $c(x)$ is equal to or less than the degree of either inputs $a(x)$ and $b(x)$, there is no need for reduction

Field Multiplication in GF(2^8)

- Given two field elements $a(x)$ and $b(x)$ with coefficients $a_i, b_i \in \text{GF}(2)$, their product is $c(x)$ is computed using a polynomial multiplication and then a polynomial reduction
- We perform a polynomial multiplication over the field GF(2), and obtain the product polynomial $c'(x)$ which is of degree at most 14

$$c'(x) = a(x) \times b(x)$$

- If degree of $c'(x)$ is less than or equal to 7, then the product $c'(x)$ is already an element of the Galois field GF(2^8), which implies $c(x) = c'(x)$

  If degree of $c'(x)$ is more than 7, then we need to reduce $c'(x)$ modulo $p(x) = x^8 + x^4 + x^3 + x + 1$ and obtain the field element $c(x) \in \text{GF}(2^8)$:

$$c(x) = c'(x) \bmod p(x)$$

Polynomial Multiplication in GF(2)

- The polynomial multiplication $c'(x) = a(x) \times b(x)$ is performed over GF(2), i.e., the coefficients are added and multiplied in GF(2)
- This implies that the product of two nonzero terms x^i and x^j is x^{i+j}
- The addition of two terms with the same power is zero: $x^i + x^i = 0$, while the different powers are not: $x^i + x^j$
- The computation of $(\texttt{a5}) \cdot (\texttt{0d}) = (1010\ 0101) \cdot (0000\ 1101)$

$$\begin{aligned} c'(x) &= (x^7 + x^5 + x^2 + 1) \times (x^3 + x^2 + 1) \\ &= x^{10} + x^8 + x^5 + x^3 + x^9 + x^7 + x^4 + x^2 + x^7 + x^5 + x^2 + 1 \\ &= x^{10} + x^9 + x^8 + \underbrace{x^7 + x^7}_{0} + \underbrace{x^5 + x^5}_{0} + x^4 + x^3 + \underbrace{x^2 + x^2}_{0} + 1 \\ &= x^{10} + x^9 + x^8 + x^4 + x^3 + 1 \end{aligned}$$

Polynomial Multiplication in GF(2)

- The polynomial multiplication over GF(2) can be performed by using a succession of AND, XOR and SHIFT operations
- Example: The computation of
  $(\texttt{a5}) \cdot (\texttt{0d}) = (1010\ 0101) \cdot (0000\ 1101)$, that is
  $c'(x) = (x^7 + x^5 + x^2 + 1) \times (x^3 + x^2 + 1)$

$$
\begin{array}{ccccccccccc}
 & & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 & & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
\cline{4-11}
 & & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & & \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & & & \\
\hline
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{array}
$$

which is obtained as: $c'(x) = x^{10} + x^9 + x^8 + x^4 + x^3 + 1$

Polynomial Reduction in GF(2)

- The second step of the multiplication operation in GF(2^8) is a polynomial reduction of $c'(x)$ modulo the irreducible polynomial $p(x)$ over the ground field GF(2)
- The reduction can be performed by hand using synthetic division of $c'(x)$ by $p(x)$, and obtaining the remainder $r(x)$ which is equal to $c(x) \in \text{GF}(2^8)$
- However, we do not need the quotient, therefore, the steps for computing the quotient can be skipped
- I will explain the synthetic division algorithm and a simplified version
- The simplified reduction algorithm can be coded as successive XOR and shift operations

Polynomial Reduction via Synthetic Division

- In the computation of $(\texttt{a5}) \cdot (\texttt{0d}) = (1010\ 0101) \cdot (0000\ 1101)$, we obtained the product $c'(x) = x^{10} + x^9 + x^8 + x^4 + x^3 + 1$
- Now, we reduce $c'(x)$ modulo $p(x) = x^8 + x^4 + x^3 + x + 1$ using synthetic division:

$$
\begin{array}{l|l}
x^{10} + x^9 + x^8 + x^4 + x^3 + 1 & x^8 + x^4 + x^3 + x + 1 \\ \cline{2-2}
x^{10} + x^6 + x^5 + x^3 + x^2 & x^2 + x + 1 \\ \cline{1-1}
x^9 + x^8 + x^6 + x^5 + x^4 + x + 2 + 1 & \\
x^9 + x^5 + x^4 + x^2 + x & \\ \cline{1-1}
x^8 + x^6 + x + 1 & \\
x^8 + x^4 + x^3 + x + 1 & \\ \cline{1-1}
x^6 + x^4 + x^3 &
\end{array}
$$

Therefore, the remainder is $x^6 + x^4 + x^3 = (0101\ 1000) = (58)$

Simplified Polynomial Reduction

- An inspection of the synthetic division algorithm shows that we perform reduction by adding shifted versions of $p(x)$ to $c'(x)$
- In the first step we have $c'(x)$, and we add $x^2 \times p(x)$ to $c'(x)$ to obtain

$$\begin{aligned} c'(x) &= c'(x) + x^2 \times p(x) \\ &= (x^{10} + x^9 + x^8 + x^4 + x^3 + 1) + x^2 \times (x^8 + x^4 + x^3 + x + 1) \\ &= x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + 1 \end{aligned}$$

- Therefore, we can simply substitute the terms x^i for $i \geq 8$ with shifted versions of the irreducible polynomial
- Also, since the highest degree term (above: x^{10}) in $c'(x)$ is to be cancelled out, there is no need to keep track of that; it is sufficient to consider the lower part of $p(x)$ which is $x^4 + x^3 + x + 1$

Simplified Polynomial Reduction

- The simplified polynomial reduction replaces an occurrence of the term x^8 in $c'(x)$ with $x^4 + x^3 + x + 1$, the lower part of $p(x)$ — similarly, x^{8+j} in $c'(x)$ is substituted with $x^j \times (x^4 + x^3 + x + 1)$
- Considering our example: $c'(x) = x^{10} + x^9 + x^8 + x^4 + x^3 + 1$, we write

$$\begin{aligned}
c'(x) &= x^{10} + x^9 + x^8 + x^4 + x^3 + 1 \\
&= x^2 \times x^8 + x \times x^8 + x^8 + x^4 + x^3 + 1 \\
&= x^2 \times (x^4 + x^3 + x + 1) + x \times (x^4 + x^3 + x + 1) + \\
&\quad +(x^4 + x^3 + x + 1) + x^4 + x^3 + 1 \\
&= (x^6 + x^5 + x^3 + x^2) + (x^5 + x^4 + x^2 + x) + \\
&\quad +(x^4 + x^3 + x + 1) + x^4 + x^3 + 1 \\
&= x^6 + x^4 + x^3
\end{aligned}$$

SHIFT-and-XOR Polynomial Reduction

- The polynomial reduction can be performed by successive XOR and SHIFT operations — this would be more amenable to programming
- The irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$ is given as $x^8 + x^4 + x^3 + x + 1$, represented as (1 0001 1011)
- The algorithm starts with the binary representation of $c'(x)$, and shifts the binary representation of the irreducible polynomial and performs XOR operations, zeroing the higher order 1s in $c'(x)$ until the upper part of $c'(x)$ (bits 8 to 13) are completely zero
- The resulting polynomial will not have any terms higher than x^7, and therefore, it would be equal to $c(x)$

SHIFT-and-XOR Polynomial Reduction

- Taking $c'(x) = x^{10} + x^9 + x^8 + x^4 + x^3 + 1 = (111\ 0001\ 1001)$ and $p(x) = (1\ 0001\ 1011)$, we perform the SHIFT-and-XOR reductions:

$$
\begin{array}{c|ccc|cccc|cccc}
 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\oplus & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & & \\
\hline
 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
\oplus & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & \\
\hline
 & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\oplus & & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
 & & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
\hline
 & & & & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0
\end{array}
$$

The result is now a polynomial of length 8 (degree at most 7) and is given as $c(x) = (0101\ 1000) = x^6 + x^4 + x^3$

Inversion in GF(2^k)

- Another important field operation for Rijndael and AES is multiplicative inversion, i.e., the computation of $b(x) = a(x)^{-1}$ in GF(2^k) such that

$$b(x) \times a(x) = 1 \bmod p(x)$$

- The inverse can be computed using the extended Euclidean algorithm, the Fermat's method, or the Itoh-Tsuji algorithm
- The extended Euclidean algorithm uses polynomial division and addition operations over the ground field GF(2)
- The Fermat's and Itoh-Tsuji methods are based on the computing the $(2^k - 2)$th power of the element $a(x)$ in the field GF(2^k) in order to obtain the inverse

Extended Euclidean Algorithm for Inverse

- Given the element $a(x)$ and the irreducible polynomial $p(x)$ such that $\mathrm{GCD}(a(x), p(x)) = 1$, the extended Euclidean algorithm computes the polynomials $s(x)$ and $t(x)$ with the property

$$a(x) \times s(x) + p(x) \times t(x) = 1$$

  From this equality we find the inverse of $a(x)$ in GF(2^k) as

$$a(x)^{-1} = s(x)$$

- Since our field of operations is GF(2^8), the irreducible polynomial is given as $p(x) = x^8 + x^4 + x^3 + x + 1$
- All polynomial division and addition operations are performed in the ground field GF(2), and the extended Euclidean Algorithm is generally more efficient for large values of k

Fermat's Method for Inversion

- The inverse can also be computed using the analogue of the Fermat's Little Theorem in the multiplicative group of GF(2^k)
- For $a \neq 0$, $a^{2^k-1} = 1$ in GF(2^k) since the group order is $2^k - 1$
- Therefore, the inverse can be computed using the identity

$$a^{-1} = a^{2^k-2}$$

  which is used in both Fermat's method and the Itoh-Tsuji method
- Considering GF(2^8), we compute the inverse of $a(x)$ in the field GF(2^8), by taking its $2^8 - 2 = 254$ power

$$a(x)^{-1} = a(x)^{2^8-2} = a(x)^{254}$$

Fermat's Method for Inversion

- We can compute $a(x)^{254}$ using the **binary method of exponentiation**
- Since $254 = (11111110)_2$, we obtain the 254th power of $a(x)$ by scanning the bits of 254 from left to right, perform a squaring for every bit while performing a multiplication by $a(x)$ if the bit is 1, as follows:

$$a \xrightarrow{s} a^2 \xrightarrow{m} a^3 \xrightarrow{s} a^6 \xrightarrow{m} a^7 \xrightarrow{s} a^{14} \xrightarrow{m} a^{15} \xrightarrow{s} a^{30}$$

$$a^{30} \xrightarrow{m} a^{31} \xrightarrow{s} a^{62} \xrightarrow{m} a^{63} \xrightarrow{s} a^{126} \xrightarrow{m} a^{127} \xrightarrow{s} a^{254}$$

- The binary method to compute the inverse of $a(x)$ in GF(2^8) requires 13 multiplications (or: 7 squarings and 6 multiplications)

Itoh-Tsuji Algorithm for Inverse

- The Itoh-Tsuji algorithm starts with

$$a^{-1} = a^{2^k-2} = (a^{2^{k-1}-1})^2$$

  and uses the factorization of $2^k - 2$ and the subsequent powers in order to compute the inverse of $a(x)$ in GF(2^k)
- Depending on whether the power j is even or odd, we can write:

$$\begin{aligned} \text{For even } (j): \quad 2^j - 1 &= (2^{\frac{j}{2}} - 1) \cdot (2^{\frac{j}{2}} + 1) \\ \text{For odd } (j): \quad 2^j - 1 &= 2 \cdot (2^{\frac{j-1}{2}} - 1) \cdot (2^{\frac{j-1}{2}} + 1) + 1 \end{aligned}$$

- The algorithm computes the powers of a using this factorization and properties: if $t = u \cdot v$, then $a^t = (a^u)^v$ and if $t = u + v$, then $a^t = a^u a^v$

Itoh-Tsuji Algorithm in GF(2^8)

- To invert in GF(2^8), we factor $2^8 - 2 = 2 \cdot (2^7 - 1)$ using the formulae:

$$\begin{aligned} 2^7 - 1 &= 2 \cdot (2^3 - 1) \cdot (2^3 + 1) + 1 = 2 \cdot 7 \cdot 9 + 1 \\ 2^3 - 1 &= 2 \cdot (2^1 - 1) \cdot (2^1 + 1) + 1 = 2 \cdot 3 + 1 \\ 2^8 - 2 &= 2 \cdot (2 \cdot 7 \cdot 9 + 1) = 2 \cdot [2 \cdot (2 \cdot 3 + 1) \cdot (2^3 + 1) + 1] \end{aligned}$$

- We now compute $a^{-1} = a^{254}$ using this factorization:

$$\begin{aligned} & a \xrightarrow{s} a^2 \xrightarrow{s} a^4 \xrightarrow{s} a^8 \xrightarrow{m} a^9 \\ & a^9 \xrightarrow{s} a^{18} \xrightarrow{m} a^{27} \xrightarrow{s} a^{54} \xrightarrow{m} a^{63} \\ & a^{63} \xrightarrow{s} a^{126} \xrightarrow{m} a^{127} \xrightarrow{s} a^{254} \end{aligned}$$

- The Itoh-Tsuji algorithm requires 11 multiplications (or: 7 squarings and 4 multiplications) which is slightly better than the binary method

Computation of SubByte in AES/Rijndael

- The SubByte step involves a table lookup operation such that an element a of the state matrix is replaced with $S(a)$
- The S table is of dimension 16×16 with 8-bit output entries
- The input to the S table is the higher a_H and lower a_L part of the element a (each of which is 4 bits, represented as 1-digit hex numbers), and output $S(a)$ is an 8-bit value, which is a 2-digit hex number
- Note that the entries of the state matrix are elements of the field GF(2^8), however, we treat them as 2-digit hex numbers in the table lookup computation of the SubByte round
- However, the SubByte also defines a nonlinear function, a mix of the field operations and the matrix-vector operations with boolean values

Table Lookup Computation of SubByte

- Consider an entry of the state matrix as (23) which is a hex value, represented as (0010 0011) in binary, and as an element of the field GF(2^8) as polynomial $a(x) = x^5 + x + 1$

$a_H \backslash a_L$	0	1	2	$\boxed{3}$	$\cdots$
0	63	7c	77	7b	$\cdots$
1	ca	82	c9	7d	$\cdots$
$\boxed{2}$	b7	fd	93	$\boxed{26}$	$\cdots$
3	04	c7	23	c3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

- We compute $S(a)$ by taking its 1-digit higher part (2) as the row index and 1-digit lower part (3) as the column index, and find

$$S(23) = 26$$

The SubByte 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

Computation of SubByte as a Nonlinear Function

- The SubByte step defines a nonlinear function which is a mix of the field operations and the matrix-vector operations with boolean entries
- Assume the input $a(x)$ is given, in order to compute $c(x) = S(a(x))$, we perform the following steps:
  Step 1: If $a(x) \neq 0$, we compute its multiplicative inverse $b(x) = a^{-1}(x)$ in the field GF(2^8); if $a(x) = 0$, we take $b(x) = 0$
  Step 2: Once $b(x)$ is available, we perform an affine transformation $c = Ab + d$ over the field GF(2), and obtain $c(x)$, where A is an 8×8 fixed matrix and d is 8×1 fixed vector, described in the next page
- The output is $c(x) = S(a(x))$

Computation of SubByte as a Nonlinear Function

- In Step 2, we apply the affine transformation to the input vector $b(x) = (b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$, and thus, compute the output vector as $c(x) = (c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0)$ using the affine transformation $c = Ab + d$ as

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- The output of this transformation is $c(x) = S(a(x))$

Computation of SubByte as a Nonlinear Function

- The computation of $S(a(x))$ involves a field inversion followed up by an affine transformation
- For field inversion, we can use any of the inversion algorithms: the extended Euclidean algorithm, the Fermat's or the Itoh-Tsuji method
- We can also create a lookup table I of size $2^8 \times 8$ bits for the inverses of all field elements, such that $I(a) = a^{-1}$; for completeness we can place $I(00) = (00)$; also note that $I(01) = (01)$ since 1 is its own inverse
- Also, $I(02) = I(x) = x^7 + x^3 + x^2 + 1$ since $x \times (x^7 + x^3 + x^2 + 1)$ equals

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

in other words $I(02) = (1000\ 1101)_2 = (\mathtt{8d})$

Computation of $S(00)$ and $S(01)$

- For $a(x) = 0$, we have $b(x) = 0$, and thus, the affine transformation $c = Ab + d$ produces the vector $c(x)$ as

$$c(x) = [1\ 1\ 0\ 0\ 0\ 1\ 1\ 0]^T = (0110\ 0011)_2 = (63)$$

  and therefore $S(0) = (63)$

- For $a(x) = 1$, we have $b(x) = 1 = (0000\ 0001)_2 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ which gives the result of the affine transformation as the sum of the first column of matrix A and the vector d:

$$\begin{aligned} c(x) &= [1\ 1\ 1\ 1\ 1\ 0\ 0\ 0]^T + [1\ 1\ 0\ 0\ 0\ 1\ 1\ 0]^T \\ &= [0\ 0\ 1\ 1\ 1\ 1\ 1\ 0] = (0111\ 1100)_2 = (7\mathrm{c}) \end{aligned}$$

  and therefore $S(01) = (7\mathrm{c})$

Computation of $S(02)$

- For $a(x) = x$, we have $b(x) = x^7 + x^3 + x^2 + 1$, and thus, we have $b(x) = (1000\ 1101)_2 = [1\ 0\ 1\ 1\ 0\ 0\ 0\ 1]^T$
- The affine transformation produces the vector $c(x)$ using

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Therefore, $c(x) = (0111\ 0111)_2$ and $S(x) = (77)$

MixColumn Operation

- The MixColumn operation multiplies a fixed 4×4 matrix with every 4×1 column vector of the state matrix
- The MixColumn matrix M in hex and polynomial representation is

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix}$$

- Given a 4×1 column vector u of the state matrix, such that each vector entry is an element of the finite field GF(2^8), we perform a matrix-vector multiplication operation Mu using field multiplications and additions to compute the new column vector of the state matrix

MixColumn Operation Example

- The following MixColumn operation example is given:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} \texttt{d4} \\ \texttt{bf} \\ \texttt{5d} \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ \texttt{e5} \end{bmatrix}$$

- In the following we show the computation of the first entry of the resulting vector, in other words, the computation of

$$[02\ 03\ 01\ 01] \begin{bmatrix} \texttt{d4} \\ \texttt{bf} \\ \texttt{5d} \\ 30 \end{bmatrix} = (04)$$

MixColumn Operation Example

- By representing the column vector $[\texttt{d4 bf 5d 30}]^T$ as a polynomial vector, we can write this MixColumn operation in polynomial representation as

$$[x \quad x+1 \quad 1 \quad 1]\left[\begin{array}{c} x^7+x^6+x^4+x^2 \\ x^7+x^5+x^4+x^3+x^2+x+1 \\ x^6+x^4+x^3+x^2+1 \\ x^5+x^4 \end{array}\right]$$

- To compute the inner product, we perform polynomial multiplications, additions, and reductions modulo $p(x)$ whenever necessary:

$$\begin{gathered} x \times (x^7+x^6+x^4+x^2)\ + \\ (x+1) \times (x^7+x^5+x^4+x^3+x^2+x+1)\ + \\ 1 \times (x^6+x^4+x^3+x^2+1)\ + \\ 1 \times (x^5+x^4) \end{gathered}$$

MixColumn Operation Example

- The first product:

$$x \times (x^7 + x^6 + x^4 + x^2)$$

  After the first product we need reduction modulo the irreducible polynomial $p(x)$ since the resulting polynomial would be of degree 8

$$\begin{aligned} x \times (x^7 + x^6 + x^4 + x^2) &= x^8 + x^7 + x^5 + x^3 \\ &= (x^4 + x^3 + x + 1) + x^7 + x^5 + x^3 \\ &= x^7 + x^5 + x^4 + x + 1 \\ &= (1011\ 0011) = (\mathtt{b3}) \end{aligned}$$

- After the polynomial multiplication, we reduced the highest degree term (which is x^8) by substituting it with $x^4 + x^3 + x + 1$, which is the lower half of the irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$

MixColumn Operation Example

- The second product:

$$(x+1)\times(x^7+x^5+x^4+x^3+x^2+x+1)$$

- After the multiplication:

$$(x+1)\times(x^7+x^5+x^4+x^3+x^2+x+1) \;=\; x^8+x^7+x^6+1$$

We need to reduce it modulo $p(x)$ since its degree is larger than 8

- By substituting x^8 with x^4+x^3+x+1, we obtain

$$x^4+x^3+x+1+x^7+x^6+1 \;=\; x^7+x^6+x^4+x^3+x$$

which is equal to $(1101\ 1010) = (\texttt{da})$

MixColumn Operation Example

- However, we do not need reductions for the third and fourth products:

$$\begin{aligned}
1 \times (x^6 + x^4 + x^3 + x^2 + 1) &= x^6 + x^4 + x^3 + x^2 + 1 &= (\mathtt{5d}) \\
1 \times (x^5 + x^4) &= x^5 + x^4 &= (\mathtt{30})
\end{aligned}$$

- Finally, adding all 4 resulting polynomials, we obtain the top entry as

$$\begin{array}{rc}
(02) \times (\mathtt{d4}) = (\mathtt{b3}) & x^7 + x^5 + x^4 + x + 1 \\
(03) \times (\mathtt{bf}) = (\mathtt{da}) & x^7 + x^6 + x^4 + x^3 + x \\
(01) \times (\mathtt{5d}) = (\mathtt{5d}) & x^6 + x^4 + x^3 + x^2 + 1 \\
(01) \times (30) = (30) & x^5 + x^4 \\
\hline
(04) & x^2
\end{array}$$

Rijndael and AES Block and Key Lengths

- Rijndael can have block lengths of 128, 192 and 256 bits, therefore, the state matrix is of dimension 4×4, 4×6, and 4×8 where the entries are the elements of GF(2^8)
- However, the AES block length is fixed at 128 bits only, and thus, the state matrix is of dimension 4×4
- On the other hand, the Rijndael and AES can have key lengths of 128, 192 and 256 bits, therefore, the key matrix is of dimension 4×4, 4×6, and 4×8 where the entries are the elements of GF(2^8)
- The number of rows in the state and key matrices for the Rijndael and AES is always 4

Rijndael and AES Block and Key Lengths

- Let N_b and N_k be the number of columns of the state matrix and key matrix, respectively
- For example, $N_b = 6$ and $N_k = 4$ gives as the state and key matrices as

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix} \qquad \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix}$$

  where each one of the state and key entries $a_{ij}, k_{lm} \in \mathrm{GF}(2^8)$
- Possible values of (N_b, N_k) for AES and Rijndael are:

$$\begin{array}{rl} \text{AES:} & (4,4),(4,6),(4,8) \\ \text{Rijndael:} & (4,4),(4,6),(4,8) \quad (6,4),(6,6),(6,8) \quad (8,4),(8,6),(8,8) \end{array}$$

Rijndael Rounds

- The number of rounds of Rijndael depends on the values of N_b and N_k:

$N_b \backslash N_k$	4	6	8
4	10	12	14
6	12	12	14
8	14	14	14

- In all cases the 0th round is just a single step of AddRoundKey, and the last round is missing the MixColumnStep, and therefore, consists of the steps: ByteSub, ShiftRow, and AddRoundKey
- For example, for $(N_b, N_k) = (6, 6)$, the 0th round is just an AddRoundKey step, while the rounds 1 through 11 consist of ByteSub, ShiftRow, MixColunm, and AddRoundKey steps, and finally the 12th round consists of ByteSub, ShiftRow, and AddRoundKey steps

Rijndael and AES Block and Key Lengths

- Due to the AddRoundKey step, the key matrix needs to match the state matrix in dimension: This works fine for the cases of $(4,4)$ for AES and Rijndael, and $(6,6),(8,8)$ for Rijndael only
- When the key matrix has more columns than the state matrix, then the unused key columns are used in the next round, which are the cases of $(4,6),(4,8)$ for AES and Rijndael, and $(6,8)$ for Rijndael only
- When the key matrix has fewer columns than the state matrix, then the key scheduling works one more step and produces the additionally needed columns for that round, and the unused key columns are used in the next round, which are the cases of $(6,4),(8,4),(8,6)$ for Rijndael only

AddRoundKey Step

- In the AddRoundKey step, the key matrix and the state matrix are added over the field GF(2^8)
- The state and the round key matrices have the same number of rows and columns, for example, they are both 4×4 as follows

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix}$$

  Here, each of $a_{ij}, k_{ij} \in \text{GF}(2^8)$ and the resulting $a_{ij} + k_{ij} \in \text{GF}(2^8)$ for $0 \leq i, j \leq 3$
- Due the property of the addition, there is no need to perform reduction

ShiftRow Transformation

- In ShiftRow the last three rows of the state matrix is shifted cyclically, but the number of shifts depend on the number of columns N_b of the state matrix
- Since for AES N_b is fixed at 4, the number of shifts for the last three rows are also fixed at 1, 2, and 3
- However, Rijndael can have different block lengths and the number of shifts is determined according to the value of N_b

N_b	c_1	c_2	c_3
4	1	2	3
6	1	2	3
8	1	3	4

Expanded Key in Key Scheduling

- The original key matrix 4, 6, and 8 columns, for key sizes 128, 192, and 256, respectively
- The **expanded key** has $N_b(1 + N_r)$ columns, where N_r is the number of rounds
- The first N_k columns of the expanded key is the columns of the original key; the remaining columns are produced recursively from the previous columns
- The key scheduling algorithm for $N_k \leq 6$ uses the same steps, while for $N_k > 6$, there is an additional SubByte step

Round Constants in Key Scheduling

- The key scheduling algorithm uses several steps, such as a columnwise cyclical shift transformation, a SubByte transformation, and an addition operation involving a variable called round constant (Rcon)
- The Rcon(i) is defined as column vectors such that

$$\text{Rcon}(i) = \begin{bmatrix} x^i \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  where the operation x^i is performed in GF(2^8), and therefore, for $i \geq 8$, it needs to be reduced modulo the irreducible polynomial
- Only the top entry of this column vector is nonzero, and the first 12 values for increasing i are 01, 02, 04, 08, 10, 20, 40, 80, 1b, 36, 6c, d8

AES Decryption

- The Rijndael is not a Feistel network, and its inverse is almost the same as the cipher — A round of the inverse cipher has the same structure
- The decryption is performed by applying the transformations in inverse order
- The AddRoundKey step is identical
- The ByteSub, MixColumn and ShiftRow steps are replaced by their inverses
- The InverseShiftRow steps performs cyclic right shift operations, instead of the cyclic left shifts in the ShiftRow step

The InverseMixColumn and the InverseSubByte

- The InverseMixColumn matrix is given as

$$\begin{bmatrix} \texttt{0e} & \texttt{0b} & \texttt{0d} & \texttt{09} \\ \texttt{09} & \texttt{0e} & \texttt{0b} & \texttt{0d} \\ \texttt{0d} & \texttt{09} & \texttt{0e} & \texttt{0b} \\ \texttt{0b} & \texttt{0d} & \texttt{09} & \texttt{0e} \end{bmatrix}$$

- In polynomial notation, this is

$$\begin{bmatrix} x^3+x^2+x & x^3+x+1 & x^3+x^2+1 & x^3+1 \\ x^3+1 & x^3+x^2+x & x^3+x+1 & x^3+x^2+1 \\ x^3+x^2+1 & x^3+1 & x^3+x^2+x & x^3+x+1 \\ x^3+x+1 & x^3+x^2+1 & x^3+1 & x^3+x^2+x \end{bmatrix}$$

The InverseMixColumn and the InverseSubByte

- The InverseMixColumn operation is not efficient as the MixColumn operation due to the fact the entries are not as as simple
- The InverseSubByte is performed over the inverse S-box
- For example, $S(00) = (63)$ implies that $S^{-1}(63) = (00)$, and similarly, $S^{-1}(7c) = (01)$ and $S^{-1}(77) = (02)$

The InverseSubByte S-Box

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	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
x	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