Chapter 3- Advanced Encryption Standard (AES)

The Security of DES

- DES has a 56-bit key, thus it can be broken in time 2^{56} by using the **brute force** attack.
- This amount of computation is definitely **feasible** today.
- DES was first broken in 96 days in 1997 by the **DESCHALL** project.
- Then it was broken in early 1998 in 41 days by the **distributed.net**project.
- A significant breakthrough came later in 1998 when it was solved in just 56hours.
    - This impressive feat was achieved via a special-purpose DES-breaking machine called *Deep Crack*(it was built by the Electronic Frontier Foundation at a cost of $250,000).
    - Additional challenges have been solved, and the latest was solved in just 22 hours and 15 minutes.

Advanced Cryptanalytic Attacks on DES

- Biham and Shamir in the late 1980s had developed a technique called *differential cryptanalysis*and used it to achieve an attack on DES in 2^{47}.
- An additional cryptanalytic attack *called linear cryptanalysis*was developed by Matsui in the early 1990s and applied to DES in 2^{43} times.
- We conclude that although using sophisticated cryptanalytic techniques it is possible to **break** DES in less time than required by a brute-force attack.

Increasing the Key Size for Block Ciphers

- The best way to avoid the brute force attack is to **double** the key length.
- Let F be a block cipher and let *k1*and *k2*be two independent keys for F.
- Then, a new block cipher with a key that is twice the length of the original one can be defined by

$$F'_{k1,k2}(x) = F_{k2}(F_{k1}(x)),$$

- If F =DES then the result is a key of size **112**.
- However, a double invocation of a block cipher does **not** provide a high enough level of security.
- We describe a "*meet-in-the middle*" attack on the double invocation method.
- Denote the length of the keys of F by *n*.
- The attack that we will describe now uses approximately 2^ntime and 2^nspace.
- The adversary is given an input/output pair (x, y) where $y=F'_{k1,k2}(x) =F_{k2}(F_{k1}(x))$, and works as follows.
    - First, it starts by building **two** lists ofpairs.
    - The first list is made up of all the pairs of the form (k1, z1) where$z1 = F_{k1}(x)$
    - The second list is made up of all the pairs of the form (k2, z2)where $z2 = F^{-1}_{k2}(y)$.
    - Notice now that there exists a value z such that $F_{k1}(x) = z = F^{-1}_{k2}(y)$, where *k1*and *k2*are the keys that the adversary is searching for.

Triple DES

- In order to **thwart** meet-in-the-middle attacks, **three** invocations of the underlying block cipher can be used.
- There are two variants that are typically used for triple invocation:
- ❖ *Variant1* - three independent keys: Choose *3 independent keys*k1, k2, k3 and compute $y = F'_{k1,k2,k3}(x) = F_{k3}(F^{-1}_{k2}(F_{k1}(x)))$.
- ❖ *Variant2* - two independent keys: Choose *2 independent* keys k1, k2 and compute $y = F'_{k1,k2}(x) = F_{k1}(F^{-1}_{k2}(F_{k1}(x)))$.

Drawbacks of 3DES are:

1- Its relatively **small** block-size
2- It is quite **slow** since it requires 3 full block cipher operations.

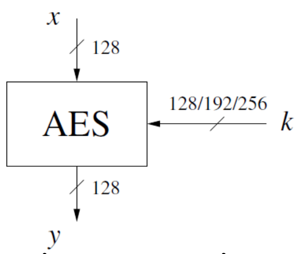
- These drawbacks have led to its recent replacement in 2001 by the *Advanced Encryption Standard* (**AES**).

AES -The Advanced Encryption Standard

- In January 1997, the *National Institute of Standards and Technology of the United States* (NIST) announced that they were seeking a new block cipher to replace the DES standard.
- The new cipher was to be called the *AdvancedEncryption Standard*, or AES for short.
- There were **15** different algorithms that were submitted from all over the world.
- These submissions included the work of many of the best cryptographers and cryptanalysts today.
- In three subsequent AES evaluation rounds, NIST and the international scientific community discussed the advantages and disadvantages of the submitted ciphers and narrowed down the number of potential candidates to **5** algorithms:
  - *Mars* by IBM Corporation,
  - *RC6* by RSA Laboratories,
  - *Rijndael*, by Joan Daemen and Vincent Rijmen,
  - *Serpent*, by Ross Anderson et al.,
  - *Twofish*, by Bruce Schneier et al.
- In October 2000 NIST announced that the winning algorithm is *Rijndael*.

AES Algorithm Overview

- Rijndael with a block length of 128 bits is known as the AES algorithm.
- In the remainder of this chapter, we only discuss the standard version of Rijndael with a block length of 128 bits.



- AES is essentially a **substitution-permutation** network,
- It **does not** have a **Feistel** structure.

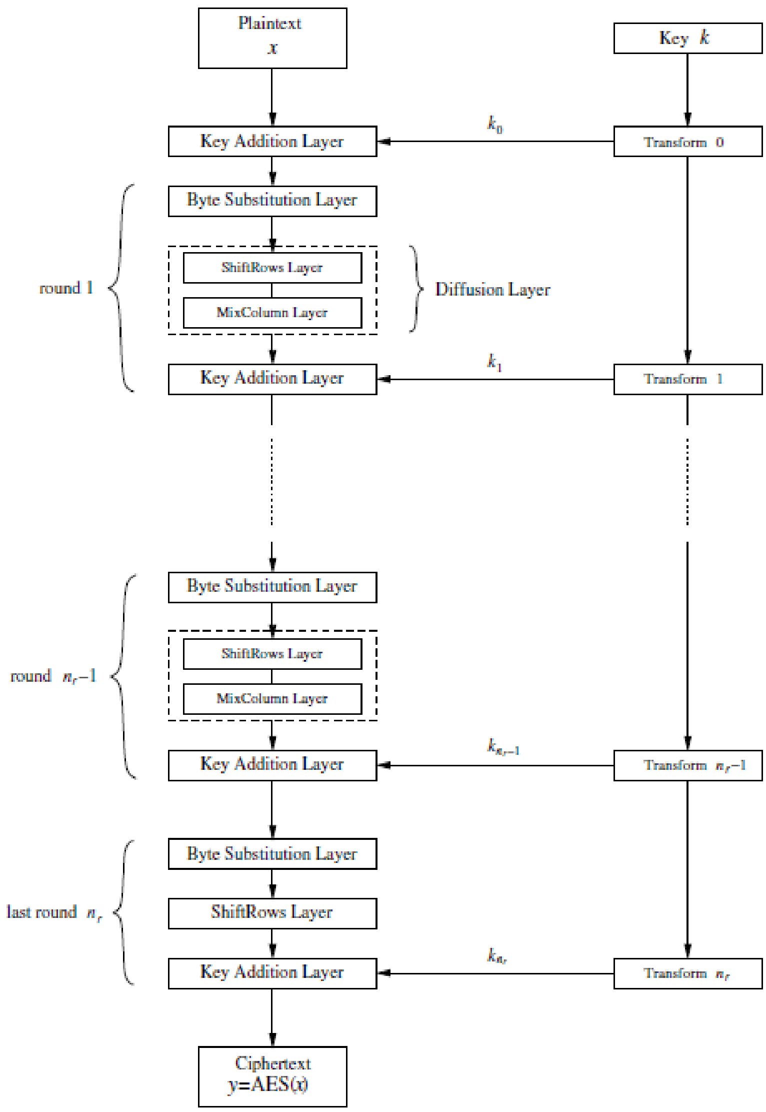
- The AES algorithm holds a 4 by 4 **array** of **bytes** called the *state*, that is initialized to the **input** to the cipher
    - **Note** that the input is 128 bits which is exactly 16 bytes.
- The number of internal rounds of the cipher is a function of the key length according to the following table:

key lengths	# rounds = n_r
128 bit	10
192 bit	12
256 bit	14

- AES encrypt **all** the 128 bits in each round, while DES encrypts only the **half** of its input in each round.
    - Thus AES has a **small** number of rounds.

Internal Structure of AES

The following figure shows the graph of AES cipher.



- In each round, the 16-byte input $A0, \ldots, A15$ is fed byte-wise into the **S-Box**.
- The 16-byte output $B0, \ldots, B15$ is **permuted** byte-wise in the **ShiftRows** layer and **mixed** by the **MixColumn** transformation.
- Finally, the 128-bit subkey k_i is **XORed** with the intermediate result.
- We note that AES is a **byte-oriented** cipher.
  - This is in contrast to DES, which makes heavy use of bit permutation and can thus be considered to have a **bit-oriented** structure.
- First we imagine that the state A (i.e., the 128-bit data path) consisting of 16 bytes $A0, A1, \ldots, A15$ is arranged in a **four-by-four byte matrix**.
- Similarly, the key bytes are arranged into a matrix with four rows and four (128-bit key):

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}

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}

Byte Substitution

- In this step, each byte of the state array is **replaced** by another byte, according to a single fixed lookup **table** S.
- This introduces ***confusion*** to the data.
- The Byte Substitution layer can be viewed as a row of 16 **parallel S-Boxes**, each with 8 input and output bits.
- Note that all 16 S-Boxes are **identical**,
  - Unlike DES where **eight different** S-Boxes are used.
- Each state byte Ai is replaced, i.e., substituted, by another byte Bi:

$$S(Ai) = Bi.$$

- The S-Box is the only *nonlinear* element of AES, i.e., it holds that $ByteSub(A) + ByteSub(B) \neq ByteSub(A+B)$ for two states A and B.
- The S-Box substitution is a **bijective** mapping, i.e., each of the $2^8 = 256$ possible input elements is **one-to-one** mapped to one output element.
  - This allows us to **uniquely reverse** the S-Box, which is needed for decryption.
- The S-Box is usually realized as a 256-by-8 bit lookup table with fixed entries, as given in the following table:

	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
x	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

Example. Let's assume the input byte to the S-Box is $A_i = (C2)_{hex}$, then the substituted value is $S((C2)_{hex}) = (25)_{hex}$.

<u>ShiftRows</u>:

- In this step, the bytes in each row of the state array are **cyclically shifted** to the left as follows:
    - The first row of the array is untouched,
    - the second row is shifted **one** place to the **left**,
    - the third row is shifted **two** places to the **left**,
    - the fourth row is shifted **three** places to the **left**.
- The purpose of the **ShiftRows** transformation is to **increase** the diffusion properties of AES.
- If the input of the **ShiftRows**sublayer is given as a state matrix $B = (B0,B1, \ldots ,B15)$:

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}

the output is the new state:

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

<u>MixColumn</u>:

- This step is a **linear** transformation which **mixes** each column of the state matrix.
- The **combination** of the **ShiftRows** and **MixColumn** layer makes it possible that after *only three* rounds every byte of the state matrix depends on all 16 plaintext bytes.
- Each 4-byte *column* is considered as a **vector** and multiplied by a fixed 4×4 matrix.
- The matrix contains *constant* entries.
- By viewing stages **ShiftRows**and **MixColumn** as a **"mixing permutation"** step (*diffusion*), we have that each round of AES has the structure of a **substitution-permutation** network.

Key Addition: In every round of AES, a 16 byte round key is **derived** from the **master** key, and is interpreted as a 4 by 4 array of bytes.Then, the key array is simply XORed with the state array. The *subkeys* are derived in the **key schedule** as follows.

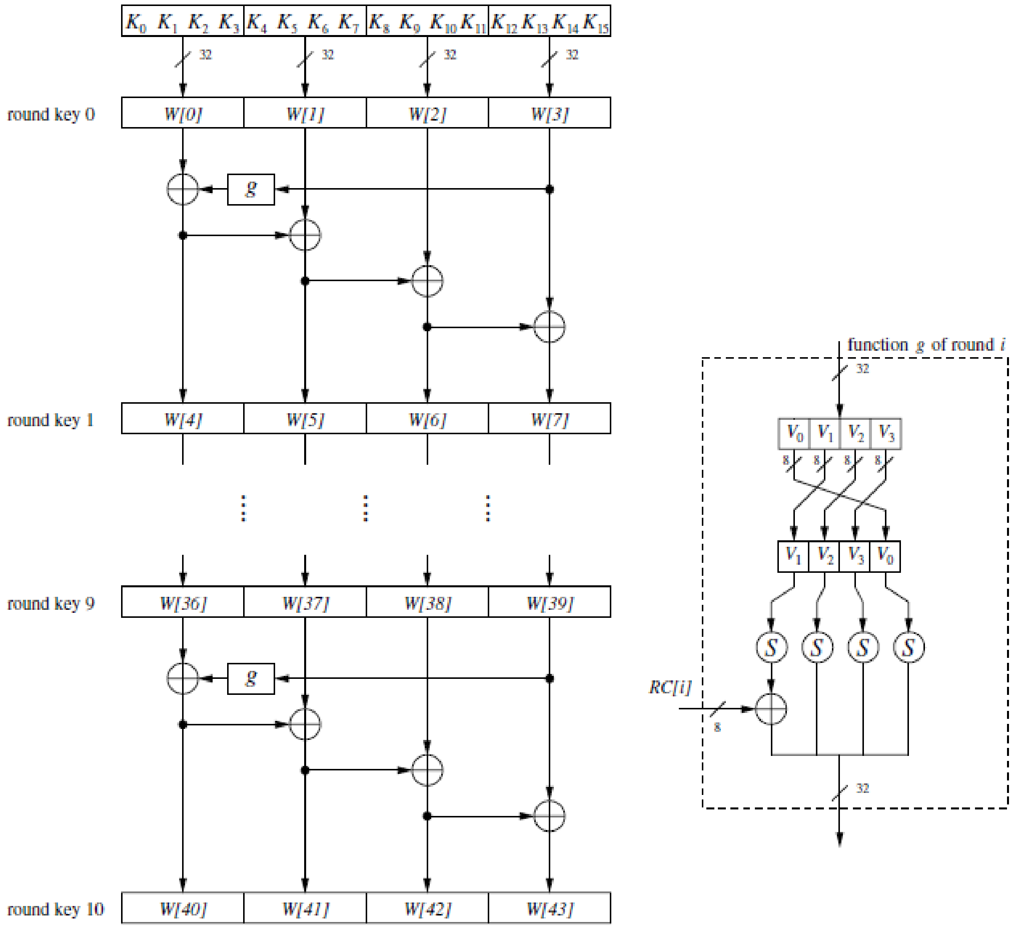
Key Schedule

- The *key schedule* takes the original input key (of length 128, 192 or 256 bit) and derives the *subkeys* used in AES.

- Note that an XOR addition of a *subkey* is used both at the input and output of AES.
    - This process is sometimes referred to as key *whitening*.
- The number of *subkeys* is equal to the number of *rounds* **plus one**, due to the key needed for key whitening in the first key addition layer.
- Thus, for the key length of 128 bits, the number of rounds is $nr = 10$, and there are 11 subkeys, each of 128 bits.

Key Schedule for 128-Bit Key AES

- The 11subkeys are stored in a key expansion array with the elements $W[0], \ldots, W[43]$.
- The subkeys are computed as depicted in the following figure.
- The elements $K0, \ldots, K15$ denote the bytes of the original AES key.



- As can be seen in the figure, the leftmost word of a subkey $W[4i]$, where $i = 1, \ldots, 10$, is computed as:

$$W[4i] = W[4(i-1)] + g(W[4i-1]).$$

- Here $g()$ is a *nonlinear function* with a **four**-byte input and output.
- The remaining three words of a subkey are computed recursively as:

$$W[4i+j] = W[4i+j-1] + W[4(i-1)+j],$$

where $i = 1, \ldots, 10$ and $j = 1, 2, 3$.

- The function $g()$:
    - **rotates** its four input bytes,

- performs a byte-wise S Box substitution,
- and adds a *round coefficient RC* to it.
  - Where RC is fixed values as:

$$
\begin{aligned}
RC[1] &= x^0 = (00000001)_2, \\
RC[2] &= x^1 = (00000010)_2, \\
RC[3] &= x^2 = (00000100)_2, \\
&\vdots \\
RC[10] &= x^9 = (00110110)_2.
\end{aligned}
$$

In general, when implementing any of the key schedules, two different approaches exist:

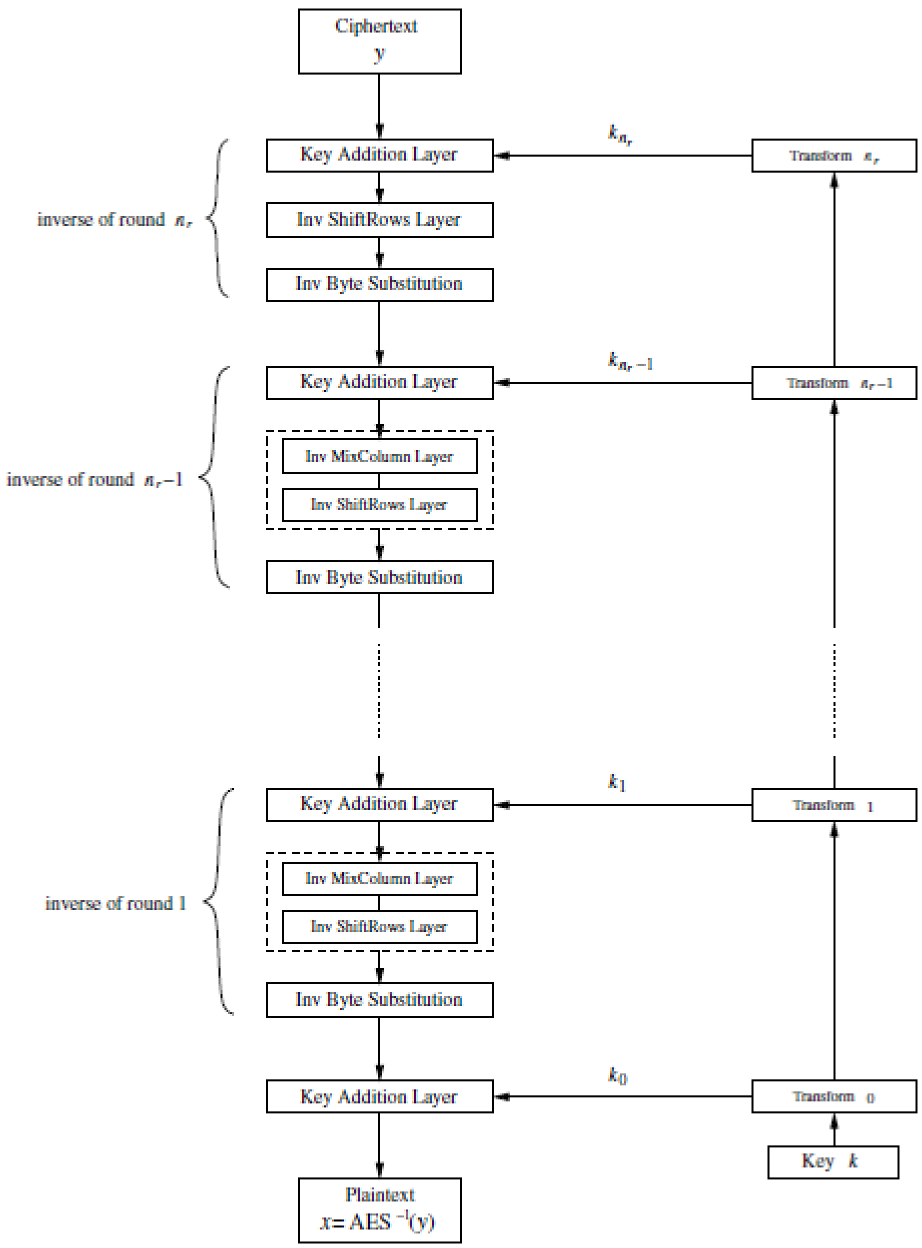
1- **Precomputation** All subkeys are expanded first into the array W.
- The encryption (decryption) of a plaintext (ciphertext) is executed afterwards.
-Note that this approach requires (nr +1) · 16 bytes of memory, e.g., 11 · 16 = 176 bytes if the key size is 128 bits.

2- **On-the-fly** A new subkey is derived for every new round during the encryption (decryption) of a plaintext (ciphertext).

- Please note that when **decrypting** ciphertexts, the **last** subkey is XORed first with the ciphertext.
  - Therefore, it is required to recursively **derive all** subkeys first and then start with the decryption of a ciphertext and the on-the-fly generation of subkeys.
  - As a result of this overhead, the decryption of a ciphertext is always slightly **slower** than the encryption of a plaintext when the on-the-fly generation of subkeys is used.

Decryption

- Because AES is not based on a Feistel network, all layers must actually be **inverted**, i.e., the **ByteSubstitution** layer becomes the **InvByteSubstitution** layer, the **ShiftRows** layer becomes the **InvShiftRows** layer, and the **MixColumn** layer becomes **InvMixColumn** layer.
- Also, we need a reversed key schedule.



Inverse MixColumn

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Inverse ShiftRows

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

Inverse Byte Substitution

x \ 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
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

Decryption Key Schedule

Since decryption round one needs the last subkey, the second decryption round needs the second-to-last subkey and so on, we need the subkey in reversed order. In practice this is mainly achieved by computing the entire key schedule first and storing all 11, 13 or 15 subkeys, depending on the number of rounds AES is using (which in turn depends on the three key lengths supported by AES). This precomputation adds usually a small latency to the decryption operation relative to encryption.