The Advanced Encryption Standard (AES)

The US government's National Institute of Standards and Technology conducted a public competition to design a replacement for DES.

- `www.nist.gov/aes`
- Sept. 1997: Call issued for AES candidate algorithms.
- Requirements:
  - Key sizes: 128, 192 and 256 bits.
  - Block size: 128 bits.
  - Efficient on both hardware and software platforms.
  - Availability on a worldwide, non-exclusive, royalty-free basis.
- Aug. 1998: 15 submissions in Round 1.
- Aug. 1999: 5 finalists.
- Dec. 2001: Rijndael is selected as the AES winner.

Rijndael is an iterated block cipher, based on a substitution-permutation network design (not a Feistel network).

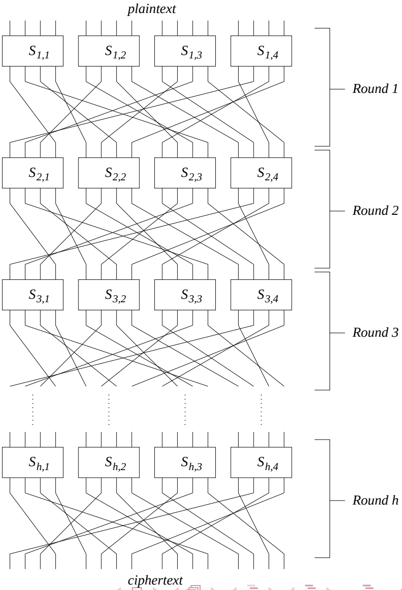
Substitution-Permutation Networks

A substitution-permutation network (SPN) is a type of iterated block cipher where a round consists of:

- A substitution operation, followed by
- A permutation operation.

Examples:

- AES
- The component function in DES



Advanced Encryption Standard

- AES is a substitution-permutation network where the "permutation" operation consists of two linear transformations (one of which is a permutation).
- All operations are byte oriented, allowing AES to be implemented efficiently on any platform.
- The block size of AES is 128 bits.
- Each round key is 128 bits.
  - A key schedule is used to generate the round keys.
- AES accepts three different key lengths. The number of rounds depends on the key length:

key length	h
128	10
192	12
256	14

AES Round Operations

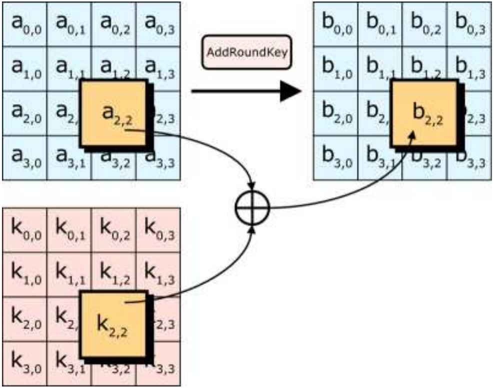
- Each round updates a variable called State which consists of a 4×4 array of bytes (note: $4 \cdot 4 \cdot 8 = 128$, the block size).
- State is initialized with the 128-bit plaintext:

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \quad \longleftarrow \quad \textit{plaintext}$$

- After h rounds are completed, one final additional round key is XOR-ed with State to produce the ciphertext (key whitening).
- The AES round function uses four operations:
  - `AddRoundKey` (key mixing)
  - `SubBytes` (S-box)
  - `ShiftRows` (permutation)
  - `MixColumns` (matrix multiplication / linear transformation)

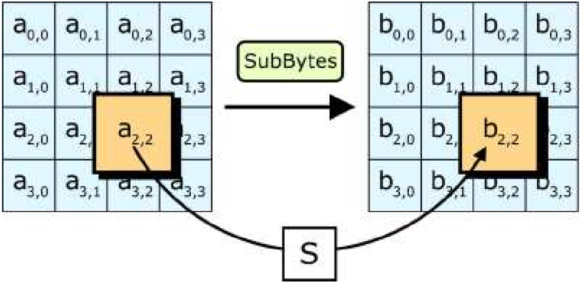
Add Round Key

Bitwise-XOR each byte of State with the corresponding byte of the round key.



Substitute Bytes

Take each byte in State and replace it with the output of the S-box.



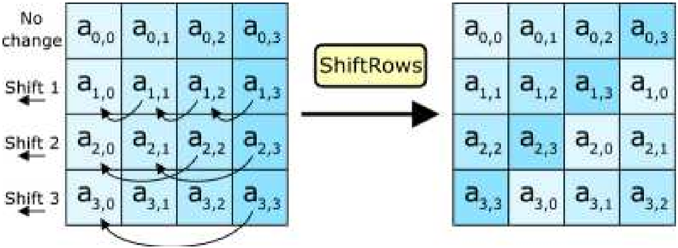
$S : \{0,1\}^8 \to \{0,1\}^8$ is a fixed and public function.

The AES S-box

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

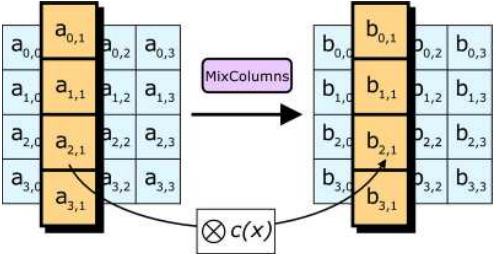
Shift Rows

Permute the bytes of State by applying a cyclic shift to each row.



Mix Columns

This step is the most mathematically complicated step in the algorithm. Multiplications take place in GF(256).



$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

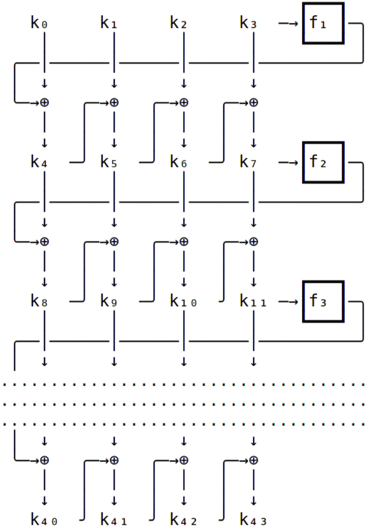
AES Encryption

- From the key k, derive $h + 1$ round keys $k_0, k_1, \ldots, k_h$ via the key schedule.
- The encryption function:

$$
\begin{array}{l}
\text{State} \leftarrow \text{plaintext} \\
\text{State} \leftarrow \text{State} \oplus k_0 \\
\text{for } i = 1 \ldots h - 1 \text{ do} \\
\qquad \text{State} \leftarrow \texttt{SubBytes}(\text{State}) \\
\qquad \text{State} \leftarrow \texttt{ShiftRows}(\text{State}) \\
\qquad \text{State} \leftarrow \texttt{MixColumns}(\text{State}) \\
\qquad \text{State} \leftarrow \text{State} \oplus k_i \\
\text{State} \leftarrow \texttt{SubBytes}(\text{State}) \\
\text{State} \leftarrow \texttt{ShiftRows}(\text{State}) \\
\text{State} \leftarrow \text{State} \oplus k_h \\
\text{ciphertext} \leftarrow \text{State}
\end{array}
$$

- Note that in the final round, `MixColumns` is not applied.

AES key schedule (for 128-bit keys)



- For 128-bit keys, AES has ten rounds, so we need eleven subkeys.
- Each k_i is a 32-bit word (viewed as a 4-byte array).
- Each group of four k_i's forms a 128-bit subkey.
- The first round subkey (k_0, k_1, k_2, k_3) equals the actual AES key.

(Diagram from `http://crypto.stackexchange.com/q/20`)

Key schedule core (for 128-bit keys)

The functions $f_i\colon \{0,1\}^{32} \to \{0,1\}^{32}$ are defined as follows:

- Left-shift the input cyclically by 8 bits.
- Apply the AES S-box to each byte.
- Bitwise XOR the left-most byte with a constant which varies by round according to the following table.

Round	constant	Round	constant
1	0x01	6	0x20
2	0x02	7	0x40
3	0x04	8	0x80
4	0x08	9	0x1B
5	0x10	10	0x36

- Output the result.

Part III

Modes of operation

Block ciphers vs. stream ciphers

Recall:

- A stream cipher is a symmetric-key encryption scheme in which each successive character of plaintext determines a single character of ciphertext.
- A block cipher is a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext.

Encrypting bulk data

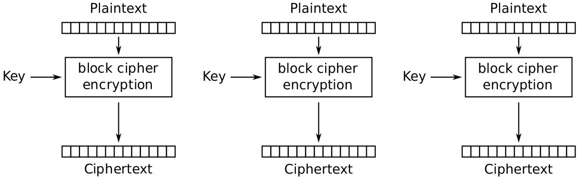
What if one needs to encrypt large quantities of data?

- With a stream cipher, just encrypt each character.
- With a block cipher, there are some complications if:
  - the input is larger than one block, or
  - the input does not fill an integer number of blocks.

To deal with these problems, we use a *mode of operation*, which means a specification for how to encrypt multiple and/or partial data blocks using a block cipher.

Electronic Codebook (ECB) mode

The obvious approach is to encrypt each ℓ bits independently, where ℓ is the block size.



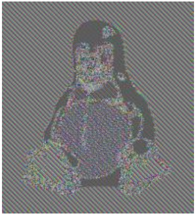
Electronic Codebook (ECB) mode encryption

(`https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`)

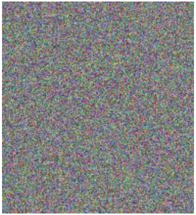
ECB mode is not semantically secure

Original

ECB mode

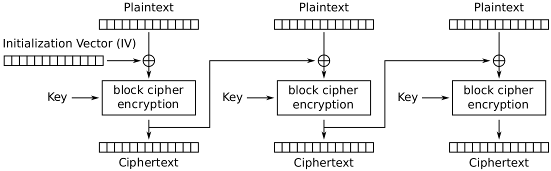


Any other mode

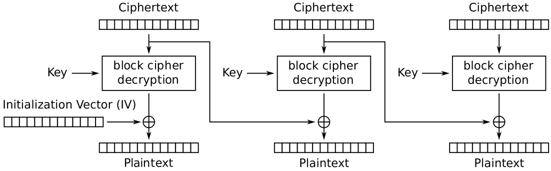


Cipher Block Chaining (CBC) mode

CBC mode: Choose a (non-secret) one-block Initialization Vector (IV) and include it as part of the ciphertext.



Cipher Block Chaining (CBC) mode encryption



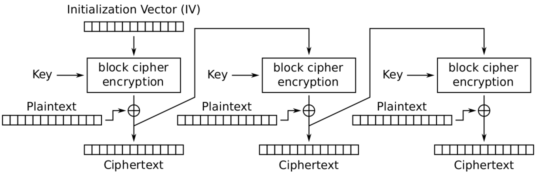
Cipher Block Chaining (CBC) mode decryption

Properties of CBC mode

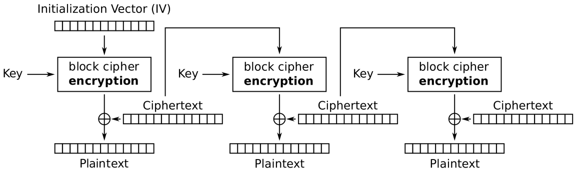
- Encryption is sequential (cannot be parallelized).
- Decryption *can* be parallelized.
- Using an IV twice under the same key invalidates semantic security. (how?)
- A small change in plaintext or IV changes all subsequent encrypted ciphertext blocks.
- A small (length-preserving) change in ciphertext changes only *two* decrypted plaintext blocks. (Active attacks are possible!)
- CBC mode does not handle partial data blocks — padding is required.

POODLE (Padding Oracle On Downgraded Legacy Encryption; published October 14, 2014) is an active attack against TLS/SSL which exploits data block padding in CBC mode.

Cipher Feedback (CFB) mode



Cipher Feedback (CFB) mode encryption

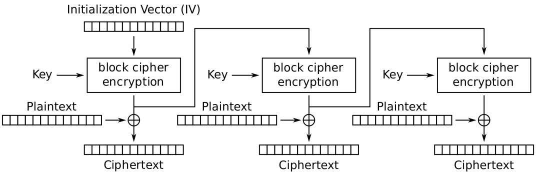


Cipher Feedback (CFB) mode decryption

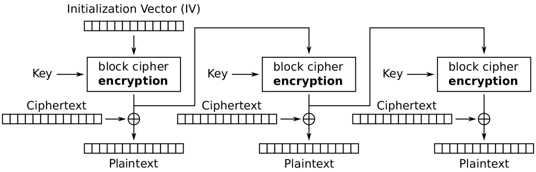
Properties of CFB mode

- The underlying block cipher is only used in **encryption** mode.
- Encryption is sequential (cannot be parallelized).
- Decryption can be parallelized.
- Using an IV twice under the same key invalidates semantic security. (Exercise: better or worse than CBC?)
- A small change in plaintext or IV changes all subsequent encrypted ciphertext blocks.
- A small (length-preserving) change in ciphertext changes two decrypted plaintext blocks. (Active attacks are possible!)
- CFB mode *can* handle partial data blocks without padding — simply transmit a partial ciphertext block.

Output Feedback (OFB) mode



Output Feedback (OFB) mode encryption



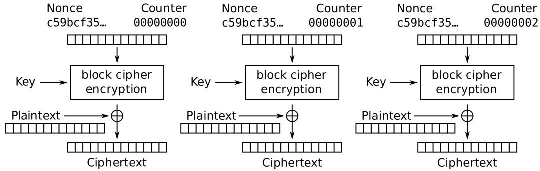
Output Feedback (OFB) mode decryption

Properties of OFB mode

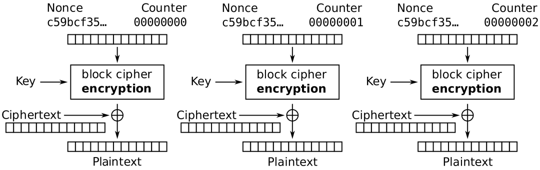
- The underlying block cipher is only used in **encryption** mode.
- Encryption cannot be parallelized, but can be pre-computed.
- Decryption cannot be parallelized.
- Using an IV twice under the same key is disastrous!
- A small change in IV changes all subsequent encrypted ciphertext blocks.
- A small (length-preserving) change in either plaintext or ciphertext produces a small change in the other.
- OFB mode can handle partial data blocks without padding — however, it is insecure in this situation, via a non-obvious attack (Davies and Parkin, 1983).

Counter (CTR) mode

Choose a nonce at random during encryption. Prepend the nonce to the ciphertext.



Counter (CTR) mode encryption

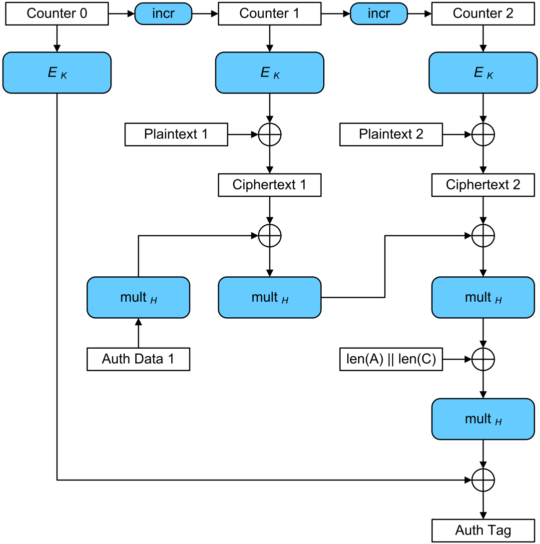


Counter (CTR) mode decryption

Properties of CTR mode

- The underlying block cipher is only used in **encryption** mode.
- Encryption and decryption are highly parallelizable.
- Using a nonce twice under the same key is disastrous!
- A small change in the nonce changes all subsequent encrypted ciphertext blocks.
- A small (length-preserving) change in either plaintext or ciphertext produces a small change in the other.
- CTR mode can handle partial data blocks without padding.

Authenticated encryption (Galois Counter Mode)



Galois Counter Mode (GCM)

- GCM ciphertexts (ignoring the authentication tag) are **identical** to counter (CTR) mode ciphertexts.
  - In particular, the last ciphertext block is truncated if the plaintext length is not an integral number of blocks.
- Authentication tags are computed in

$$\mathrm{GF}(2^{128}) = \mathbb{F}_2[x]/(x^{128} + x^7 + x^2 + x + 1).$$

- Hence, GCM requires a 128-bit block size (e.g. AES).
- "Auth Data 1" is a 128-bit block of authenticated unencrypted data, viewed as an element of $\mathrm{GF}(2^{128})$.
  - More than one such block is supported, but only one is shown.
- H is defined as $H = E_k(0^{128}) = E_k(\mathbf{0}) \in \mathrm{GF}(2^{128})$. Computing H requires knowledge of the key.
- Computing authentication tags can be parallelized using field arithmetic.