Block-Cipher Modes

Modern symmetric cryptosystems tend to be block ciphers, where the algorithm simply specifies how to encrypt/decrypt a b bit block of data. The most simple way to utilize one of these algorithms is to separate the data into several blocks and encrypt each block, exactly as the algorithm specifies. However, there are other "modes" independent to the encryption algorithm, that can be used.

In fact, the first idea, is a mode called Electronic Code Book(ECB), which is the least secure of all the modes one could use.

This list below shows the different modes one can use to implement a symmetric cryptosystem when sending messages of many b-bit blocks:

1) Electronic Code Book(ECB): This is the typical manner in which block cipher are executed. Simply grab n bits of the plaintext at a time, where n is the block size, and then encrypt them using the algorithm, and output the corresponding ciphertext. Repeat for each subsequent block.

for i=1 to n:
$C_i = E(K, P_i)$

here P_i stands for the ith block of the plaintext and C_i stands for the ith block of the cipher text.

2) Cipher Block Chaining(CBC) mode:

C_0 = IV (initialization vector)
i = 1
for i=1 to n:
$C_i = E(K, C_{i-1} \oplus P_i)$

What this means is that we used the ciphertext from the previous block, XOR it with the next Plaintext block, and then encrypt this result, instead of just encrypting the plaintext block itself.

An advantage here is that most of the ciphertext doesn't even represent plaintext encrypted. The disadvantage here is that everything has to be done sequentially, because the computation for C_{10} can't begin until C_9 is known, etc.

3) Output Feedback Mode(OFB):

This is a stream cipher, which means that we obtain the ciphertext by XORing the plaintext with a keystream. The keystream is generated with the key and the block cipher system being used.

$K_0 = IV$
$i = 1$
for i=1 to n {
  $K_i = E(K, K_{i-1})$
  $C_i = P_i \oplus K_i$
}

Here, we don't even encrypt the message!!! Instead, we start with a random initial vector, and then encrypt it over and over again with our K to produce effectively random bits that are based on the previous set of bits generated. Then, those random bits are XORed with the plaintext block.

This has the advantage that the values **of K_i can be precomputed in advance**, so that this algorithm could be run in parallel.

4) Cipher Feedback Mode(CFB):

This is also a stream cipher, except that the keystream is generated by encrypting the ciphertext instead of the plaintext.

$C_0 = IV$
for i=1 to n {
  $K_i = E(K, C_{i-1})$
  $C_i = P_i \oplus K_i$
}

In this one, we encrypt previous ciphertexts to create future "key strings" which we XOR with the message. This one must be done sequentially, since K_i is based on C_{i-1}.

5) Counter Mode (CTR)

Here, the counter array stores known, prechosen values, so that each block could be encrypted in parallel. It could just store, 1, 2, 3, etc. Or, the initial counter value could be provided and then future counter values could be based on it, but if this were the case, then the counter values would have to be determined sequentially.

for i=1 to n {
  $K_i = E(K, counter[i])$
  $C_i = M_i \oplus K_i$
}