Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'des_modes.7ossl' command

$ man des_modes.7ossl

DES_MODES(7ossl) OpenSSL DES_MODES(7ossl)

NAME

des_modes - the variants of DES and other crypto algorithms of OpenSSL

DESCRIPTION

Several crypto algorithms for OpenSSL can be used in a number of modes.

Those are used for using block ciphers in a way similar to stream

ciphers, among other things.

OVERVIEW

Electronic Codebook Mode (ECB)

Normally, this is found as the function algorithm_ecb_encrypt().

? 64 bits are enciphered at a time.

? The order of the blocks can be rearranged without detection.

? The same plaintext block always produces the same ciphertext block

(for the same key) making it vulnerable to a 'dictionary attack'.

? An error will only affect one ciphertext block.

Cipher Block Chaining Mode (CBC)

Normally, this is found as the function algorithm_cbc_encrypt(). Be aware that des_cbc_encrypt() is not really DES CBC (it does not update the IV); use des_ncbc_encrypt() instead.

? a multiple of 64 bits are enciphered at a time.

? The CBC mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable.

? The chaining operation makes the ciphertext blocks dependent on the current and all preceding plaintext blocks and therefore blocks can not be rearranged.

? The use of different starting variables prevents the same plaintext enciphering to the same ciphertext.

? An error will affect the current and the following ciphertext blocks.

Cipher Feedback Mode (CFB)

Normally, this is found as the function algorithm_cfb_encrypt().

? a number of bits (j) <= 64 are enciphered at a time.

? The CFB mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable.

? The chaining operation makes the ciphertext variables dependent on the current and all preceding variables and therefore j-bit variables are chained together and can not be rearranged.

? The use of different starting variables prevents the same plaintext enciphering to the same ciphertext.

? The strength of the CFB mode depends on the size of k (maximal if j == k). In my implementation this is always the case.

? Selection of a small value for j will require more cycles through the encipherment algorithm per unit of plaintext and thus cause greater processing overheads.

? Only multiples of j bits can be enciphered.

? An error will affect the current and the following ciphertext variables.

Output Feedback Mode (OFB)

Normally, this is found as the function algorithm_ofb_encrypt().

? a number of bits (j) <= 64 are enciphered at a time.

? The OFB mode produces the same ciphertext whenever the same plaintext enciphered using the same key and starting variable. More over, in the OFB mode the same key stream is produced when the same key and start variable are used. Consequently, for security reasons a specific start variable should be used only once for a given key.

? The absence of chaining makes the OFB more vulnerable to specific attacks.

? The use of different start variables values prevents the same plaintext enciphering to the same ciphertext, by producing different key streams.

? Selection of a small value for j will require more cycles through the encipherment algorithm per unit of plaintext and thus cause greater

processing overheads.

? Only multiples of j bits can be enciphered.

? OFB mode of operation does not extend ciphertext errors in the resultant plaintext output. Every bit error in the ciphertext causes only one bit to be in error in the deciphered plaintext.

? OFB mode is not self-synchronizing. If the two operation of encipherment and decipherment get out of synchronism, the system needs to be re-initialized.

? Each re-initialization should use a value of the start variable different from the start variable values used before with the same key. The reason for this is that an identical bit stream would be produced each time from the same parameters. This would be susceptible to a 'known plaintext' attack.

Triple ECB Mode

Normally, this is found as the function algorithm_ecb3_encrypt().

? Encrypt with key1, decrypt with key2 and encrypt with key3 again.

? As for ECB encryption but increases the key length to 168 bits. There are theoretic attacks that can be used that make the effective key length 112 bits, but this attack also requires 2^56 blocks of memory, not very likely, even for the NSA.

? If both keys are the same it is equivalent to encrypting once with just one key.

? If the first and last key are the same, the key length is 112 bits. There are attacks that could reduce the effective key strength to

only slightly more than 56 bits, but these require a lot of memory.

? If all 3 keys are the same, this is effectively the same as normal ecb mode.

Triple CBC Mode

Normally, this is found as the function algorithm_ede3_cbc_encrypt().

? Encrypt with key1, decrypt with key2 and then encrypt with key3.

? As for CBC encryption but increases the key length to 168 bits with the same restrictions as for triple ecb mode.

NOTES

This text was been written in large parts by Eric Young in his original documentation for SSLeay, the predecessor of OpenSSL. In turn, he attributed it to:

AS 2805.5.2
Australian Standard
Electronic funds transfer - Requirements for interfaces,
Part 5.2: Modes of operation for an n-bit block cipher algorithm
Appendix A

SEE ALSO

BF_encrypt(3), DES_crypt(3)

COPYRIGHT

<https://www.openssl.org/source/license.html>.