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 'RSA_private_decrypt.3ossl' command

$ man RSA_private_decrypt.3ossl

RSA_PUBLIC_ENCRYPT(3ossl) OpenSSL RSA_PUBLIC_ENCRYPT(3ossl)

NAME

RSA_public_encrypt, RSA_private_decrypt - RSA public key cryptography

SYNOPSIS

#include <openssl/rsa.h>

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining OPENSSL_API_COMPAT with a suitable version value, see openssl_user_macros(7):

int RSA_public_encrypt(int flen, const unsigned char *from,
unsigned char *to, RSA *rsa, int padding);

int RSA_private_decrypt(int flen, const unsigned char *from,
unsigned char *to, RSA *rsa, int padding);

DESCRIPTION

Both of the functions described on this page are deprecated. Applications should instead use EVP_PKEY_encrypt_init_ex(3), EVP_PKEY_encrypt(3), EVP_PKEY_decrypt_init_ex(3) and EVP_PKEY_decrypt(3).

RSA_public_encrypt() encrypts the flen bytes at from (usually a session key) using the public key rsa and stores the ciphertext in to. to must point to RSA_size(rsa) bytes of memory.

padding denotes one of the following modes:

RSA_PKCS1_PADDING

PKCS #1 v1.5 padding. This currently is the most widely used mode. However, it is highly recommended to use RSA_PKCS1_OAEP_PADDING in new applications. SEE WARNING BELOW.

RSA_PKCS1_OAEP_PADDING

EME-OAEP as defined in PKCS #1 v2.0 with SHA-1, MGF1 and an empty encoding parameter. This mode is recommended for all new applications.

RSA_NO_PADDING

Raw RSA encryption. This mode should only be used to implement cryptographically sound padding modes in the application code. Encrypting user data directly with RSA is insecure.

flen must not be more than RSA_size(rsa) - 11 for the PKCS #1 v1.5 based padding modes, not more than RSA_size(rsa) - 42 for RSA_PKCS1_OAEP_PADDING and exactly RSA_size(rsa) for RSA_NO_PADDING. When a padding mode other than RSA_NO_PADDING is in use, then RSA_public_encrypt() will include some random bytes into the ciphertext and therefore the ciphertext will be different each time, even if the plaintext and the public key are exactly identical. The returned ciphertext in to will always be zero padded to exactly RSA_size(rsa) bytes. to and from may overlap.

RSA_private_decrypt() decrypts the flen bytes at from using the private

key rsa and stores the plaintext in to. flen should be equal to RSA_size(rsa) but may be smaller, when leading zero bytes are in the ciphertext. Those are not important and may be removed, but RSA_public_encrypt() does not do that. to must point to a memory section large enough to hold the maximal possible decrypted data (which is equal to RSA_size(rsa) for RSA_NO_PADDING, RSA_size(rsa) - 11 for the PKCS #1 v1.5 based padding modes and RSA_size(rsa) - 42 for RSA_PKCS1_OAEP_PADDING). padding is the padding mode that was used to encrypt the data. to and from may overlap.

RETURN VALUES

RSA_public_encrypt() returns the size of the encrypted data (i.e., RSA_size(rsa)). RSA_private_decrypt() returns the size of the recovered plaintext. A return value of 0 is not an error and means only that the plaintext was empty.

On error, -1 is returned; the error codes can be obtained by ERR_get_error(3).

WARNINGS

Decryption failures in the RSA_PKCS1_PADDING mode leak information which can potentially be used to mount a Bleichenbacher padding oracle attack. This is an inherent weakness in the PKCS #1 v1.5 padding design. Prefer RSA_PKCS1_OAEP_PADDING.

CONFORMING TO

SSL, PKCS #1 v2.0

SEE ALSO

ERR_get_error(3), RAND_bytes(3), RSA_size(3)

HISTORY

Both of these functions were deprecated in OpenSSL 3.0.

COPYRIGHT