

NAME

krb5_crypto_getblocksize, **krb5_crypto_getconfoundersize**, **krb5_crypto_getenctype**,
krb5_crypto_getpadsize, **krb5_crypto_overhead**, **krb5_decrypt**, **krb5_decrypt_EncryptedData**,
krb5_decrypt_ivec, **krb5_decrypt_ticket**, **krb5_encrypt**, **krb5_encrypt_EncryptedData**,
krb5_encrypt_ivec, **krb5_encrypt_disable**, **krb5_encrypt_keysize**, **krb5_encrypt_to_string**,
krb5_encrypt_valid, **krb5_get_wrapped_length**, **krb5_string_to_enctype** - encrypt and decrypt data, set
and get encryption type parameters

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

krb5_error_code

krb5_encrypt(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*, *void *data*, *size_t len*,
*krb5_data *result*);

krb5_error_code

krb5_encrypt_EncryptedData(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*, *void *data*,
size_t len, *int kvno*, *EncryptedData *result*);

krb5_error_code

krb5_encrypt_ivec(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*, *void *data*, *size_t len*,
*krb5_data *result*, *void *ivec*);

krb5_error_code

krb5_decrypt(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*, *void *data*, *size_t len*,
*krb5_data *result*);

krb5_error_code

krb5_decrypt_EncryptedData(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*,
*EncryptedData *e*, *krb5_data *result*);

krb5_error_code

krb5_decrypt_ivec(*krb5_context context*, *krb5_crypto crypto*, *unsigned usage*, *void *data*, *size_t len*,
*krb5_data *result*, *void *ivec*);

krb5_error_code

krb5_decrypt_ticket(*krb5_context context*, *Ticket *ticket*, *krb5_keyblock *key*, *EncTicketPart *out*,

krb5_flags flags);

krb5_error_code

krb5_crypto_getblocksize(*krb5_context context*, *size_t *blocksize*);

krb5_error_code

krb5_crypto_getenctype(*krb5_context context*, *krb5_crypto crypto*, *krb5_enctype *enctype*);

krb5_error_code

krb5_crypto_getpadsizesize(*krb5_context context*, *size_t, *padsizesize*");

krb5_error_code

krb5_crypto_getconfoundersize(*krb5_context context*, *krb5_crypto crypto*, *size_t, *confoundersize*");

krb5_error_code

krb5_encrypt_keysize(*krb5_context context*, *krb5_enctype type*, *size_t *keysize*);

krb5_error_code

krb5_crypto_overhead(*krb5_context context*, *size_t, *padsizesize*");

krb5_error_code

krb5_string_to_enctype(*krb5_context context*, *const char *string*, *krb5_enctype *etype*);

krb5_error_code

krb5_enctype_to_string(*krb5_context context*, *krb5_enctype etype*, *char **string*);

krb5_error_code

krb5_enctype_valid(*krb5_context context*, *krb5_enctype etype*);

void

krb5_enctype_disable(*krb5_context context*, *krb5_enctype etype*);

size_t

krb5_get_wrapped_length(*krb5_context context*, *krb5_crypto crypto*, *size_t data_len*);

DESCRIPTION

These functions are used to encrypt and decrypt data.

krb5_encrypt_ivec() puts the encrypted version of *data* (of size *len*) in *result*. If the encryption type supports using derived keys, *usage* should be the appropriate key-usage. *ivec* is a pointer to a initial IV,

it is modified to the end IV at the end of the round. *Ivec* should be the size of *If*. If NULL is passed in, the default IV is used. **krb5_encrypt()** does the same as **krb5_encrypt_ivec()** but with *ivec* being NULL. **krb5_encrypt_EncryptedData()** does the same as **krb5_encrypt()**, but it puts the encrypted data in a *EncryptedData* structure instead. If *kvno* is not zero, it will be put in the (optional) *kvno* field in the *EncryptedData*.

krb5_decrypt_ivec(), **krb5_decrypt()**, and **krb5_decrypt_EncryptedData()** works similarly.

krb5_decrypt_ticket() decrypts the encrypted part of *ticket* with *key*. **krb5_decrypt_ticket()** also verifies the timestamp in the ticket, invalid flag and if the KDC haven't verified the transited path, the transit path.

krb5_enctype_keysize(), **krb5_crypto_getconfoundersize()**, **krb5_crypto_getblocksize()**, **krb5_crypto_getenctype()**, **krb5_crypto_getpadsizesize()**, **krb5_crypto_overhead()** all returns various (sometimes) useful information from a crypto context. **krb5_crypto_overhead()** is the combination of **krb5_crypto_getconfoundersize**, **krb5_crypto_getblocksize** and **krb5_crypto_getpadsizesize** and return the maximum overhead size.

krb5_enctype_to_string() converts a encryption type number to a string that can be printable and stored. The strings returned should be freed with **free(3)**.

krb5_string_to_enctype() converts a encryption type strings to a encryption type number that can use used for other Kerberos crypto functions.

krb5_enctype_valid() returns 0 if the encrypt is supported and not disabled, otherwise and error code is returned.

krb5_enctype_disable() (globally, for all contextes) disables the *enctype*.

krb5_get_wrapped_length() returns the size of an encrypted packet by *crypto* of length *data_len*.

SEE ALSO

krb5_create_checksum(3), **krb5_crypto_init(3)**