

**NAME**

Heimdal Kerberos 5 cryptography functions -

**Functions**

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_enctype\_valid** (krb5\_context context, krb5\_enctype etype)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_cksumtype\_to\_enctype** (krb5\_context context, krb5\_cksumtype ctype, krb5\_enctype \*etype)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_encrypt\_iov\_ivec** (krb5\_context context, krb5\_crypto crypto, unsigned usage, **krb5\_crypto\_iov** \*data, int num\_data, void \*ivec)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_decrypt\_iov\_ivec** (krb5\_context context, krb5\_crypto crypto, unsigned usage, **krb5\_crypto\_iov** \*data, unsigned int num\_data, void \*ivec)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_create\_checksum\_iov** (krb5\_context context, krb5\_crypto crypto, unsigned usage, **krb5\_crypto\_iov** \*data, unsigned int num\_data, krb5\_cksumtype \*type)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_verify\_checksum\_iov** (krb5\_context context, krb5\_crypto crypto, unsigned usage, **krb5\_crypto\_iov** \*data, unsigned int num\_data, krb5\_cksumtype \*type)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_init** (krb5\_context context, const krb5\_keyblock \*key, krb5\_enctype etype, krb5\_crypto \*crypto)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_destroy** (krb5\_context context, krb5\_crypto crypto)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_getblocksize** (krb5\_context context, krb5\_crypto crypto, size\_t \*blocksize)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_getenctype** (krb5\_context context, krb5\_crypto crypto, krb5\_enctype \*enctype)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_getpadsizesize** (krb5\_context context, krb5\_crypto crypto, size\_t \*padsizesize)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_getconfoundersize** (krb5\_context context, krb5\_crypto crypto, size\_t \*confoundersize)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_enctype\_disable** (krb5\_context context, krb5\_enctype enctype)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_enctype\_enable** (krb5\_context context, krb5\_enctype enctype)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_allow\_weak\_crypto** (krb5\_context context, krb5\_boolean enable)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_random\_to\_key** (krb5\_context context, krb5\_enctype type, const void \*data, size\_t size, krb5\_keyblock \*key)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_crypto\_fx\_cf2** (krb5\_context context, const krb5\_crypto crypto1, const krb5\_crypto crypto2, krb5\_data \*pepper1, krb5\_data \*pepper2,

krb5\_etype etype, krb5\_keyblock \*res)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_generate\_subkey\_extended**

(krb5\_context context, const krb5\_keyblock \*key, krb5\_etype etype, krb5\_keyblock \*\*subkey)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_keyblock\_zero** (krb5\_keyblock \*keyblock)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_free\_keyblock\_contents** (krb5\_context context, krb5\_keyblock \*keyblock)

KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL **krb5\_free\_keyblock** (krb5\_context context, krb5\_keyblock \*keyblock)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_keyblock\_contents**

(krb5\_context context, const krb5\_keyblock \*inblock, krb5\_keyblock \*to)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_copy\_keyblock** (krb5\_context context,

const krb5\_keyblock \*inblock, krb5\_keyblock \*\*to)

KRB5\_LIB\_FUNCTION krb5\_etype KRB5\_LIB\_CALL **krb5\_keyblock\_get\_etype** (const krb5\_keyblock \*block)

KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL **krb5\_keyblock\_init** (krb5\_context context, krb5\_etype etype, const void \*data, size\_t size, krb5\_keyblock \*key)

## Detailed Description

### Function Documentation

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_allow\_weak\_crypto** (krb5\_context context, krb5\_boolean enable)

Enable or disable all weak encryption types

#### Parameters:

*context* Kerberos 5 context

*enable* true to enable, false to disable

#### Returns:

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_cksumtype\_to\_etype** (krb5\_context context, krb5\_cksumtype ctype, krb5\_etype \* etype)

Return the corresponding encryption type for a checksum type.

#### Parameters:

*context* Kerberos context

*ctype* The checksum type to get the result etype for

*etype* The returned encryption, when the matching etype is not found, etype is set to ETYPE\_NULL.

**Returns:**

Return an error code for an failure or 0 on success.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_keyblock (krb5\_context context, const krb5\_keyblock \* inblock, krb5\_keyblock \*\* to)**

Copy a keyblock, free the output keyblock with **krb5\_free\_keyblock()**.

**Parameters:**

*context* a Kerberos 5 context

*inblock* the key to copy

*to* the output key.

**Returns:**

0 on success or a Kerberos 5 error code

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_copy\_keyblock\_contents (krb5\_context context, const krb5\_keyblock \* inblock, krb5\_keyblock \* to)**

Copy a keyblock, free the output keyblock with **krb5\_free\_keyblock\_contents()**.

**Parameters:**

*context* a Kerberos 5 context

*inblock* the key to copy

*to* the output key.

**Returns:**

0 on success or a Kerberos 5 error code

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_create\_checksum\_iov (krb5\_context context, krb5\_crypto crypto, unsigned usage, krb5\_crypto\_iov \* data, unsigned int num\_data, krb5\_cksumtype \* type)**

Create a Kerberos message checksum.

**Parameters:**

*context* Kerberos context

*crypto* Kerberos crypto context

*usage* Key usage for this buffer

*data* array of buffers to process

*num\_data* length of array

*type* output data

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_destroy (krb5\_context context, krb5\_crypto crypto)**

Free a crypto context created by **krb5\_crypto\_init()**.

**Parameters:**

*context* Kerberos context

*crypto* crypto context to free

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_fx\_cf2 (krb5\_context context, const krb5\_crypto crypto1, const krb5\_crypto crypto2, krb5\_data \* pepper1, krb5\_data \* pepper2, krb5\_enctype enctype, krb5\_keyblock \* res)**

The FX-CF2 key derivation function, used in FAST and preauth framework.

**Parameters:**

*context* Kerberos 5 context

*crypto1* first key to combine

*crypto2* second key to combine

*pepper1* factor to combine with first key to garante uniqueness

*pepper2* factor to combine with second key to garante uniqueness

*enctype* the encryption type of the resulting key

*res* allocated key, free with **krb5\_free\_keyblock\_contents()**

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_getblocksize (krb5\_context context, krb5\_crypto crypto, size\_t \* blocksize)**

Return the blocksize used algorithm referenced by the crypto context

**Parameters:**

*context* Kerberos context

*crypto* crypto context to query

*blocksize* the resulting blocksize

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_getconfoundersize (krb5\_context context, krb5\_crypto crypto, size\_t \* confoundersize)**

Return the confounder size used by the crypto context

**Parameters:**

*context* Kerberos context

*crypto* crypto context to query

*confoundersize* the returned confounder size

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_getenctype (krb5\_context context, krb5\_crypto crypto, krb5\_enctype \* enctype)**

Return the encryption type used by the crypto context

**Parameters:**

*context* Kerberos context

*crypto* crypto context to query

*enctype* the resulting encryption type

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_getpadsizesize (krb5\_context context, krb5\_crypto crypto, size\_t \* padsizesize)**

Return the padding size used by the crypto context

**Parameters:**

*context* Kerberos context

*crypto* crypto context to query

*padsizesize* the return padding size

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_crypto\_init (krb5\_context context,**

**const krb5\_keyblock \* key, krb5\_etype etype, krb5\_crypto \* crypto)**

Create a crypto context used for all encryption and signature operation. The encryption type to use is taken from the key, but can be overridden with the etype parameter. This can be useful for encryptions types which is compatible (DES for example).

To free the crypto context, use **krb5\_crypto\_destroy()**.

**Parameters:**

*context* Kerberos context  
*key* the key block information with all key data  
*etype* the encryption type  
*crypto* the resulting crypto context

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_decrypt\_iov\_ivec (krb5\_context context, krb5\_crypto crypto, unsigned usage, krb5\_crypto\_iov \* data, unsigned int num\_data, void \* ivec)**

Inline decrypt a Kerberos message.

**Parameters:**

*context* Kerberos context  
*crypto* Kerberos crypto context  
*usage* Key usage for this buffer  
*data* array of buffers to process  
*num\_data* length of array  
*ivec* initial cbc/cts vector

**Returns:**

Return an error code or 0.

1. KRB5\_CRYPTO\_TYPE\_HEADER 2. one KRB5\_CRYPTO\_TYPE\_DATA and array [0,...] of KRB5\_CRYPTO\_TYPE\_SIGN\_ONLY in any order, however the receiver have to aware of the order. KRB5\_CRYPTO\_TYPE\_SIGN\_ONLY is commonly used unencrypted protocol headers and trailers. The output data will be of same size as the input data or shorter.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_encrypt\_iov\_ivec (krb5\_context context, krb5\_crypto crypto, unsigned usage, krb5\_crypto\_iov \* data, int num\_data, void \* ivec)**

Inline encrypt a kerberos message

**Parameters:**

*context* Kerberos context  
*crypto* Kerberos crypto context  
*usage* Key usage for this buffer  
*data* array of buffers to process  
*num\_data* length of array  
*ivec* initial cbc/cts vector

**Returns:**

Return an error code or 0.

Kerberos encrypted data look like this:

1. KRB5\_CRYPTO\_TYPE\_HEADER 2. array [1,...] KRB5\_CRYPTO\_TYPE\_DATA and array [0,...] KRB5\_CRYPTO\_TYPE\_SIGN\_ONLY in any order, however the receiver have to aware of the order. KRB5\_CRYPTO\_TYPE\_SIGN\_ONLY is commonly used headers and trailers. 3. KRB5\_CRYPTO\_TYPE\_PADDING, at least on padsize long if padsize > 1 4. KRB5\_CRYPTO\_TYPE\_TRAILER

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_etype\_disable (krb5\_context context, krb5\_etype etype)**

Disable encryption type

**Parameters:**

*context* Kerberos 5 context  
*etype* encryption type to disable

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_etype\_enable (krb5\_context context, krb5\_etype etype)**

Enable encryption type

**Parameters:**

*context* Kerberos 5 context  
*etype* encryption type to enable

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_etype\_valid (krb5\_context context, krb5\_etype etype)**

Check if a enctype is valid, return 0 if it is.

**Parameters:**

*context* Kerberos context

*etype* enctype to check if its valid or not

**Returns:**

Return an error code for an failure or 0 on success (enctype valid).

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_free\_keyblock (krb5\_context context, krb5\_keyblock \* keyblock)**

Free a keyblock, also zero out the content of the keyblock, uses **krb5\_free\_keyblock\_contents()** to free the content.

**Parameters:**

*context* a Kerberos 5 context

*keyblock* keyblock to free, NULL is valid argument

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_free\_keyblock\_contents (krb5\_context context, krb5\_keyblock \* keyblock)**

Free a keyblock's content, also zero out the content of the keyblock.

**Parameters:**

*context* a Kerberos 5 context

*keyblock* keyblock content to free, NULL is valid argument

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_generate\_subkey\_extended (krb5\_context context, const krb5\_keyblock \* key, krb5\_etype etype, krb5\_keyblock \*\* subkey)**

Generate subkey, from keyblock

**Parameters:**

*context* kerberos context

*key* session key

*etype* encryption type of subkey, if ETYPE\_NULL, use key's enctype

*subkey* returned new, free with **krb5\_free\_keyblock()**.

**Returns:**

0 on success or a Kerberos 5 error code



**KRB5\_LIB\_FUNCTION krb5\_etype KRB5\_LIB\_CALL krb5\_keyblock\_get\_etype (const krb5\_keyblock \* block)**

Get encryption type of a keyblock.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_keyblock\_init (krb5\_context context, krb5\_etype type, const void \* data, size\_t size, krb5\_keyblock \* key)**

Fill in 'key' with key data of type 'etype' from 'data' of length 'size'. Key should be freed using **krb5\_free\_keyblock\_contents()**.

**Returns:**

0 on success or a Kerberos 5 error code

**KRB5\_LIB\_FUNCTION void KRB5\_LIB\_CALL krb5\_keyblock\_zero (krb5\_keyblock \* keyblock)**

Zero out a keyblock

**Parameters:**

*keyblock* keyblock to zero out

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_random\_to\_key (krb5\_context context, krb5\_etype type, const void \* data, size\_t size, krb5\_keyblock \* key)**

Converts the random bytestring to a protocol key according to Kerberos crypto frame work. It may be assumed that all the bits of the input string are equally random, even though the entropy present in the random source may be limited.

**Parameters:**

*context* Kerberos 5 context

*type* the etype resulting key will be of

*data* input random data to convert to a key

*size* size of input random data, at least **krb5\_etype\_keysize()** long

*key* key, output key, free with **krb5\_free\_keyblock\_contents()**

**Returns:**

Return an error code or 0.

**KRB5\_LIB\_FUNCTION krb5\_error\_code KRB5\_LIB\_CALL krb5\_verify\_checksum\_iov (krb5\_context context, krb5\_crypto crypto, unsigned usage, krb5\_crypto\_iov \* data, unsigned int num\_data, krb5\_cksumtype \* type)**

Verify a Kerberos message checksum.

**Parameters:**

*context* Kerberos context  
*crypto* Kerberos crypto context  
*usage* Key usage for this buffer  
*data* array of buffers to process  
*num\_data* length of array  
*type* return checksum type if not NULL

**Returns:**

Return an error code or 0.