NAME

krb5_checksum, **krb5_checksum_disable**, **krb5_checksum_is_collision_proof**, **krb5_checksum_is_keyed**, **krb5_checksumsize**, **krb5_cksumtype_valid**, **krb5_copy_checksum**, **krb5_create_checksum**, **krb5_crypto_get_checksum_type krb5_free_checksum**, **krb5_free_checksum_contents**, **krb5_hmac**, **krb5_verify_checksum** - creates, handles and verifies checksums

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

typedef Checksum krb5_checksum;

void
krb5_checksum_disable(*krb5_context context, krb5_cksumtype type*);

krb5_boolean
krb5_checksum_is_collision_proof(*krb5_context context, krb5_cksumtype type*);

krb5_boolean
krb5_checksum_is_keyed(*krb5_context context, krb5_cksumtype type*);

krb5_error_code
krb5_cksumtype_valid(*krb5_context context, krb5_cksumtype ctype*);

krb5_error_code
krb5_checksumsize(*krb5_context context, krb5_cksumtype type, size_t *size*);

krb5_error_code
krb5_create_checksum(*krb5_context context, krb5_crypto crypto, krb5_key_usage usage, int type, void *data, size_t len, Checksum *result*);

krb5_error_code
krb5_verify_checksum(*krb5_context context, krb5_crypto crypto, krb5_key_usage usage, void *data, size_t len, Checksum *cksum*);

krb5_error_code
krb5_crypto_get_checksum_type(*krb5_context context, krb5_crypto crypto, krb5_cksumtype *type*);

void
krb5_free_checksum(*krb5_context context, krb5_checksum *cksum*);

void
krb5_free_checksum_contents(*krb5_context context, krb5_checksum *cksum*);

krb5_error_code
krb5_hmac(*krb5_context context, krb5_cksumtype cktype, const void *data, size_t len, unsigned usage, krb5_keyblock *key, Checksum *result*);

krb5_error_code
krb5_copy_checksum(*krb5_context context, const krb5_checksum *old, krb5_checksum **new*);

DESCRIPTION

The krb5_checksum structure holds a Kerberos checksum. There is no component inside krb5_checksum that is directly referable.

The functions are used to create and verify checksums. **krb5_create_checksum**() creates a checksum of the specified data, and puts it in *result*. If *crypto* is NULL, *usage_or_type* specifies the checksum type to use; it must not be keyed. Otherwise *crypto* is an encryption context created by **krb5_crypto_init**(), and *usage_or_type* specifies a key-usage.

krb5_verify_checksum() verifies the *checksum* against the provided data.

krb5_checksum_is_collision_proof() returns true is the specified checksum is collision proof (that it's very unlikely that two strings has the same hash value, and that it's hard to find two strings that has the same hash). Examples of collision proof checksums are MD5, and SHA1, while CRC32 is not.

krb5_checksum_is_keyed() returns true if the specified checksum type is keyed (that the hash value is a function of both the data, and a separate key). Examples of keyed hash algorithms are HMAC-SHA1-DES3, and RSA-MD5-DES. The "plain" hash functions MD5, and SHA1 are not keyed.

krb5_crypto_get_checksum_type() returns the checksum type that will be used when creating a checksum for the given *crypto* context. This function is useful in combination with **krb5_checksumsize**() when you want to know the size a checksum will use when you create it.

krb5_cksumtype_valid() returns 0 or an error if the checksumtype is implemented and not currently disabled in this kerberos library.

krb5_checksumsize() returns the size of the outdata of checksum function.

krb5_copy_checksum() returns a copy of the checksum **krb5_free_checksum**() should use used to free the *new* checksum.

krb5_free_checksum() free the checksum and the content of the checksum.

krb5_free_checksum_contents() frees the content of checksum in *cksum*.

krb5_hmac() calculates the HMAC over *data* (with length *len*) using the keyusage *usage* and keyblock *key*. Note that keyusage is not always used in checksums.

krb5_checksum_disable globally disables the checksum type.

SEE ALSO

krb5_crypto_init(3), krb5_c_encrypt(3), krb5_encrypt(3)