NAME

crypto_session - state used for symmetric cryptographic services

SYNOPSIS

#include <opencrypto/cryptodev.h>

*struct auth_hash **
crypto_auth_hash(*const struct crypto_session_params *csp*);

*struct enc_xform **
crypto_cipher(*const struct crypto_session_params *csp*);

*const struct crypto_session_params **
crypto_get_params(*crypto_session_t cses*);

int
crypto_newsession(*crypto_session_t *cses*, *const struct crypto_session_params *csp*, *int crid*);

int
crypto_freesession(*crypto_session_t cses*);

DESCRIPTION

Symmetric cryptographic operations in the kernel are associated with cryptographic sessions. Sessions hold state shared across multiple requests. Active sessions are associated with a single cryptographic driver.

The *crypto_session_t* type represents an opaque reference to an active session. Session objects are allocated and managed by the cryptographic framework.

New sessions are created by **crypto_newsession**(). *csp* describes various parameters associated with the new session such as the algorithms to use and any session-wide keys. *crid* can be used to request either a specific cryptographic driver or classes of drivers. For the latter case, *crid* should be set to a mask of the following values:

CRYPTOCAP_F_HARDWARE Request hardware drivers. Hardware drivers do not use the host CPU to perform operations. Typically, a separate co-processor performs the operations asynchronously.

CRYPTOCAP_F_SOFTWARE Request software drivers. Software drivers use the host CPU to perform operations. The kernel includes a simple, yet portable implementation of each supported algorithm in the cryptosoft(4) driver. Additional software drivers may also be available on architectures which provide instructions designed to accelerate cryptographic operations.

If both hardware and software drivers are requested, hardware drivers are preferred over software drivers. Accelerated software drivers are preferred over the baseline software driver. If multiple hardware drivers are available, the framework will distribute sessions across these drivers in a round-robin fashion.

On success, **crypto_newsession**() saves a reference to the newly created session in *cses*.

crypto_freesession() is used to free the resources associated with the session *cses*.

crypto_auth_hash() returns a structure describing the baseline software implementation of an authentication algorithm requested by *csp*. If *csp* does not specify an authentication algorithm, or requests an invalid algorithm, NULL is returned.

crypto_cipher() returns a structure describing the baseline software implementation of an encryption algorithm requested by *csp*. If *csp* does not specify an encryption algorithm, or requests an invalid algorithm, NULL is returned.

crypto_get_params() returns a pointer to the session parameters used by *cses*.

Session Parameters

Session parameters are used to describe the cryptographic operations performed by cryptographic requests. Parameters are stored in an instance of *struct crypto_session_params*. When initializing parameters to pass to **crypto_newsession**(), the entire structure should first be zeroed. Needed fields should then be set leaving unused fields as zero. This structure contains the following fields:

csp_mode Type of operation to perform. This field must be set to one of the following:

CSP_MODE_COMPRESS Compress or decompress request payload.

The compression algorithm is specified in *csp_cipher_alg*.

CSP_MODE_CIPHER Encrypt or decrypt request payload.

The encryption algorithm is specified in *csp_cipher_alg*.

CSP_MODE_DIGEST Compute or verify a digest, or hash, of request payload.

The authentication algorithm is specified in *csp_auth_alg*.

CSP_MODE_AEAD Authenticated encryption with additional data. Decryption operations require the digest, or tag, and fail if it does not match.

The AEAD algorithm is specified in *csp_cipher_alg*.

CSP_MODE_ETA Encrypt-then-Authenticate. In this mode, encryption operations encrypt the payload and then compute an authentication digest over the request additional authentication data followed by the encrypted payload. Decryption operations fail without decrypting the data if the provided digest does not match.

The encryption algorithm is specified in *csp_cipher_alg* and the authentication algorithm is specified in *csp_auth_alg*.

csp_flags A mask of optional driver features. Drivers will only attach to a session if they support all of the requested features.

CSP_F_SEPARATE_OUTPUT Support requests that use separate input and output buffers. Sessions with this flag set permit requests with either a single buffer that is modified in-place, or requests with separate input and output buffers. Sessions without this flag only permit requests with a single buffer that is modified in-place.

CSP_F_SEPARATE_AAD Support requests that use a separate buffer for AAD rather than providing AAD as a region in the input buffer. Sessions with this flag set permit requests with AAD passed in either in a region of the input buffer or in a single, virtually-contiguous buffer. Sessions without this flag only permit requests with AAD passed in as a region in the input buffer.

CSP_F_ESN Support requests that use a separate buffer for IPsec

ESN (Extended Sequence Numbers).

Sessions with this flag set permit requests with IPsec ESN passed in special buffer. It is required for IPsec ESN support of encrypt and authenticate mode where the high-order 32 bits of the sequence number are appended after the Next Header (RFC 4303).

csp_ivlen If either the cipher or authentication algorithms require an explicit initialization vector (IV) or nonce, this specifies the length in bytes. All requests for a session use the same IV length.

csp_cipher_alg Encryption or compression algorithm.

csp_cipher_klen Length of encryption or decryption key in bytes. All requests for a session use the same key length.

csp_cipher_key Pointer to encryption or decryption key. If all requests for a session use request-specific keys, this field should be left as NULL. This pointer and associated key must remain valid for the duration of the crypto session.

csp_auth_alg Authentication algorithm.

csp_auth_klen Length of authentication key in bytes. If the authentication algorithm does not use a key, this field should be left as zero.

csp_auth_key Pointer to the authentication key. If all requests for a session use request-specific keys, this field should be left as NULL. This pointer and associated key must remain valid for the duration of the crypto session.

csp_auth_mlen The length in bytes of the digest. If zero, the full length of the digest is used. If non-zero, the first *csp_auth_mlen* bytes of the digest are used.

RETURN VALUES

crypto_newsession() returns a non-zero value if an error occurs or zero on success.

crypto_auth_hash() and **crypto_cipher**() return NULL if the request is valid or a pointer to a structure on success.

SEE ALSO

crypto(7), crypto(9), crypto_request(9)

BUGS

The current implementation of **crypto_freesession** does not provide a way for the caller to know that there are no other references to the keys stored in the session's associated parameters. This function should probably sleep until any in-flight cryptographic operations associated with the session are completed.