NAME

crypto_driver - interface for symmetric cryptographic drivers

SYNOPSIS

#include <opencrypto/cryptodev.h>

void
crypto_copyback(*struct cryptop *crp, int off, int size, const void *src*);

void
crypto_copydata(*struct cryptop *crp, int off, int size, void *dst*);

void
crypto_done(*struct cryptop *crp*);

int32_t
crypto_get_driverid(*device_t dev, size_t session_size, int flags*);

*void **
crypto_get_driver_session(*crypto_session_t crypto_session*);

void
crypto_read_iv(*struct cryptop *crp, void *iv*);

int
crypto_unblock(*uint32_t driverid, int what*);

int
crypto_unregister_all(*uint32_t driverid*);

int
CRYPTODEV_FREESESSION(*device_t dev, crypto_session_t crypto_session*);

int
CRYPTODEV_NEWSESSION(*device_t dev, crypto_session_t crypto_session,*
*const struct crypto_session_params *csp*);

int
CRYPTODEV_PROBESESSION(*device_t dev, const struct crypto_session_params *csp*);

int
CRYPTODEV_PROCESS(*device_t dev, struct cryptop *crp, int flags*);

void
hmac_init_ipad(*struct auth_hash *axf, const char *key, int klen, void *auth_ctx*);

void
hmac_init_opad(*struct auth_hash *axf, const char *key, int klen, void *auth_ctx*);

DESCRIPTION

Symmetric cryptographic drivers process cryptographic requests submitted to sessions associated with the driver.

Cryptographic drivers call **crypto_get_driverid**() to register with the cryptographic framework. *dev* is the device used to service requests. The **CRYPTODEV**() methods are defined in the method table for the device driver attached to *dev*. *session_size* specifies the size of a driver-specific per-session structure allocated by the cryptographic framework. *flags* is a bitmask of properties about the driver. Exactly one of CRYPTOCAP_F_SOFTWARE or CRYPTOCAP_F_HARDWARE must be specified. CRYPTOCAP_F_SOFTWARE should be used for drivers which process requests using host CPUs. CRYPTOCAP_F_HARDWARE should be used for drivers which process requests on separate co-processors. CRYPTOCAP_F_SYNC should be set for drivers which process requests synchronously in **CRYPTODEV_PROCESS**(). CRYPTOCAP_F_ACCEL_SOFTWARE should be set for software drivers which use accelerated CPU instructions. **crypto_get_driverid**() returns an opaque driver id.

crypto_unregister_all() unregisters a driver from the cryptographic framework. If there are any pending operations or open sessions, this function will sleep. *driverid* is the value returned by an earlier call to **crypto_get_driverid**().

When a new session is created by **crypto_newsession**(), **CRYPTODEV_PROBESESSION**() is invoked by the cryptographic framework on each active driver to determine the best driver to use for the session. This method should inspect the session parameters in *csp*. If a driver does not support requests described by *csp*, this method should return an error value. If the driver does support requests described by *csp*, it should return a negative value. The framework prefers drivers with the largest negative value, similar to DEVICE_PROBE(9). The following values are defined for non-error return values from this method:

CRYPTODEV_PROBE_HARDWARE	The driver processes requests via a co-processor.
CRYPTODEV_PROBE_ACCEL_SOFTWARE	The driver processes requests on the host CPU using optimized instructions such as AES-NI.

CRYPTODEV_PROBE_SOFTWARE The driver processes requests on the host CPU.

This method should not sleep.

Once the framework has chosen a driver for a session, the framework invokes the **CRYPTODEV_NEWSESSION**() method to initialize driver-specific session state. Prior to calling this method, the framework allocates a per-session driver-specific data structure. This structure is initialized with zeroes, and its size is set by the *session_size* passed to **crypto_get_driverid**(). This method can retrieve a pointer to this data structure by passing *crypto_session* to **crypto_get_driver_session**(). Session parameters are described in *csp*.

This method should not sleep.

CRYPTODEV_FREESESSION() is invoked to release any driver-specific state when a session is destroyed. The per-session driver-specific data structure is explicitly zeroed and freed by the framework after this method returns. If a driver requires no additional tear-down steps, it can leave this method undefined.

This method should not sleep.

CRYPTODEV_PROCESS() is invoked for each request submitted to an active session. This method can either complete a request synchronously or schedule it to be completed asynchronously, but it must not sleep.

If this method is not able to complete a request due to insufficient resources such as a full command queue, it can defer the request by returning ERESTART. The request will be queued by the framework and retried once the driver releases pending requests via **crypto_unblock**(). Any requests submitted to sessions belonging to the driver will also be queued until **crypto_unblock**() is called.

If a driver encounters errors while processing a request, it should report them via the *crp_etype* field of *crp* rather than returning an error directly.

flags may be set to CRYPTO_HINT_MORE if there are additional requests queued for this driver. The driver can use this as a hint to batch completion interrupts. Note that these additional requests may be from different sessions.

crypto_get_driver_session() returns a pointer to the driver-specific per-session data structure for the session *crypto_session*. This function can be used in the **CRYPTODEV_NEWSESSION**(), **CRYPTODEV_PROCESS**(), and **CRYPTODEV_FREESESSION**() callbacks.

crypto_copydata() copies *size* bytes out of the input buffer for *crp* into a local buffer pointed to by *dst*. The bytes are read starting at an offset of *off* bytes in the request's input buffer.

crypto_copyback() copies *size* bytes from the local buffer pointed to by *src* into the output buffer for *crp*. The bytes are written starting at an offset of *off* bytes in the request's output buffer.

crypto_read_iv() copies the IV or nonce for *crp* into the local buffer pointed to by *iv*.

A driver calls **crypto_done**() to mark the request *crp* as completed. Any errors should be set in *crp_etype* prior to calling this function.

If a driver defers a request by returning ERESTART from CRYPTO_PROCESS, the framework will queue all requests for the driver until the driver calls **crypto_unblock**() to indicate that the temporary resource shortage has been relieved. For example, if a driver returns ERESTART due to a full command ring, it would invoke **crypto_unblock**() from a command completion interrupt that makes a command ring entry available. *driverid* is the value returned by **crypto_get_driverid**(). *what* indicates which types of requests the driver is able to handle again:

CRYPTO_SYMQ indicates that the driver is able to handle symmetric requests passed to **CRYPTODEV_PROCESS**().

hmac_init_ipad() prepares an authentication context to generate the inner hash of an HMAC. *axf* is a software implementation of an authentication algorithm such as the value returned by **crypto_auth_hash**(). *key* is a pointer to a HMAC key of *klen* bytes. *auth_ctx* points to a valid authentication context for the desired algorithm. The function initializes the context with the supplied key.

hmac_init_opad() is similar to **hmac_init_ipad**() except that it prepares an authentication context to generate the outer hash of an HMAC.

RETURN VALUES

crypto_apply() returns the return value from the caller-supplied callback function.

crypto_contiguous_subsegment() returns a pointer to a contiguous segment or NULL.

crypto_get_driverid() returns a driver identifier on success or -1 on error.

crypto_unblock(), **crypto_unregister_all**(), **CRYPTODEV_FREESESSION**(), **CRYPTODEV_NEWSESSION**(), and **CRYPTODEV_PROCESS**() return zero on success or an error on failure.

CRYPTODEV_PROBESESSION() returns a negative value on success or an error on failure.

SEE ALSO

crypto(7), crypto(9), crypto_buffer(9), crypto_request(9), crypto_session(9)