NAME

```
crypto_buffer - symmetric cryptographic request buffers
```

SYNOPSIS

```
#include <opencrypto/cryptodev.h>
int
crypto_apply(struct cryptop *crp, int off, int len, int (*f)(void *, void *, u_int), void *arg);
int
crypto_apply_buf(struct crypto_buffer *cb, int off, int len, int (*f)(void *, void *, u_int), void *arg);
void *
crypto_buffer_contiguous_subsegment(struct crypto_buffer *cb, size_t skip, size_t len);
size_t
crypto_buffer_len(struct crypto_buffer *cb);
void *
crypto_contiguous_subsegment(struct cryptop *crp, size_t skip, size_t len);
void
crypto_cursor_init(struct crypto_buffer_cursor *cc, const struct crypto_buffer *cb);
void
crypto_cursor_advance(struct crypto_buffer_cursor *cc, size_t amount);
void
crypto_cursor_copyback(struct crypto_buffer_cursor *cc, int size, const void *src);
void
crypto_cursor_copydata(struct crypto_buffer_cursor *cc, int size, void *dst);
void
crypto_cursor_copydata_noadv(struct crypto_buffer_cursor *cc, int size, void *dst);
void *
crypto_cursor_segment(struct crypto_buffer_cursor *cc, size_t *len);
void
```

crypto_cursor_copy(const struct crypto_buffer_cursor *fromc, struct crypto_buffer_cursor *toc);

bool

CRYPTO_HAS_OUTPUT_BUFFER(struct cryptop *crp);

DESCRIPTION

Symmetric cryptographic requests use data buffers to describe the data to be modified. Requests can either specify a single data buffer whose contents are modified in place, or requests may specify separate data buffers for input and output. *struct crypto_buffer* provides an abstraction that permits cryptographic requests to operate on different types of buffers. *struct crypto_cursor* allows cryptographic drivers to iterate over a data buffer.

CRYPTO_HAS_OUTPUT_BUFFER() returns true if *crp* uses separate buffers for input and output and false if *crp* uses a single buffer.

crypto_buffer_len() returns the length of data buffer *cb* in bytes.

crypto_apply_buf() invokes a caller-supplied function to a region of the data buffer cb. The function f is called one or more times. For each invocation, the first argument to f is the value of arg passed to $crypto_apply_buf$ (). The second and third arguments to f are a pointer and length to a segment of the buffer mapped into the kernel. The function is called enough times to cover the len bytes of the data buffer which starts at an offset off. If any invocation of f returns a non-zero value, $crypto_apply_buf$ () immediately returns that value without invoking f on any remaining segments of the region, otherwise $crypto_apply_buf$ () returns the value from the final call to f. $crypto_apply$ () invokes the callback f on a region of the input data buffer for crp.

crypto_buffer_contiguous_subsegment() attempts to locate a single, virtually-contiguous segment of the data buffer *cb*. The segment must be *len* bytes long and start at an offset of *skip* bytes. If a segment is found, a pointer to the start of the segment is returned. Otherwise, NULL is returned. **crypto_contiguous_subsegment**() attempts to locate a single, virtually-contiguous segment in the input data buffer for *crp*.

Data Buffers

Data buffers are described by an instance of *struct crypto buffer*. The *cb_type* member contains the type of the data buffer. The following types are supported:

CRYPTO_BUF_NONE An invalid buffer. Used to mark the output buffer when a crypto request uses a single data buffer.

CRYPTO BUF CONTIG An array of bytes mapped into the kernel's address space.

CRYPTO_BUFFER(9) FreeBSD Kernel Developer's Manual CRYPTO_BUFFER(9)

CRYPTO BUF UIO A scatter/gather list of kernel buffers as described in uio(9).

CRYPTO BUF MBUF A chain of network memory buffers as described in mbuf(9).

CRYPTO_BUF_SINGLE_MBUF

A single network memory buffer as described in mbuf(9).

CRYPTO BUF VMPAGE A scatter/gather list of vm page t structures describing pages in the

kernel's address space. This buffer type is only available if

CRYPTO_HAS_VMPAGE is true.

The structure also contains the following type-specific fields:

cb_buf A pointer to the start of a CRYPTO_BUF_CONTIG data buffer.

cb_buf_len The length of a CRYPTO_BUF_CONTIG data buffer

cb_mbuf A pointer to a struct mbuf for CRYPTO_BUF_MBUF and

CRYPTO_BUF_SINGLE_MBUF.

cb_uio A pointer to a *struct uio* for CRYPTO_BUF_UIO.

cb_vm_page A pointer to an array of struct vm_page for CRYPTO_BUF_VMPAGE.

cb_vm_page_len The total amount of data included in the cb_vm_page array, in bytes.

cb_vm_page_offset Offset in bytes in the first page of cb_vm_page where valid data begins.

Cursors

Cursors provide a mechanism for iterating over a data buffer. They are primarily intended for use in software drivers which access data buffers via virtual addresses.

crypto_cursor_init() initializes the cursor *cc* to reference the start of the data buffer *cb*.

crypto_cursor_advance() advances the cursor *amount* bytes forward in the data buffer.

crypto_cursor_copyback() copies *size* bytes from the local buffer pointed to by *src* into the data buffer associated with *cc*. The bytes are written to the current position of *cc*, and the cursor is then advanced by *size* bytes.

crypto_cursor_copydata() copies *size* bytes out of the data buffer associated with *cc* into a local buffer pointed to by *dst*. The bytes are read from the current position of *cc*, and the cursor is then advanced by *size* bytes.

crypto_cursor_copydata_noadv() is similar to **crypto_cursor_copydata**() except that it does not change the current position of *cc*.

crypto_cursor_segment() returns the start of the virtually-contiguous segment at the current position of *cc*. The length of the segment is stored in *len*.

RETURN VALUES

crypto_apply() and **crypto_apply_buf**() return the return value from the caller-supplied callback function.

crypto_buffer_contiguous_subsegment(), crypto_contiguous_subsegment(), and crypto_cursor_segment() return a pointer to a contiguous segment or NULL.

crypto_buffer_len() returns the length of a buffer in bytes.

crypto_cursor_seglen() returns the length in bytes of a contiguous segment.

crypto_cursor_copy() makes a deep copy of the cursor *fromc*. The two copies do not share any state and can thus be used independently.

CRYPTO_HAS_OUTPUT_BUFFER() returns true if the request uses a separate output buffer.

SEE ALSO

ipsec(4), crypto(7), bus_dma(9), crypto(9), crypto_driver(9), crypto_request(9), crypto_session(9), mbuf(9), uio(9)

HISTORY

The **crypto_buffer** functions first appeared in FreeBSD 13.

AUTHORS

The **crypto_buffer** functions and this manual page were written by John Baldwin < jhb@FreeBSD.org>.