Code Examples

Code Example For Symmetric Key Cipher Operation

This code encrypts some data with AES-256-XTS. For sake of example, all inputs are random bytes, the encryption is done in-place, and it's assumed the code is running in a context where it can sleep.

```
static int test_skcipher(void)
{
        struct crypto_skcipher *tfm = NULL;
        struct skcipher_request *req = NULL;
        u8 *data = NULL;
        const size_t datasize = 512; /* data size in bytes */
        struct scatterlist sg;
        DECLARE_CRYPTO_WAIT(wait);
        u8 iv[16];  /* AES-256-XTS takes a 16-byte IV */
        u8 key[64]; /* AES-256-XTS takes a 64-byte key */
        int err;

        /*
         * Allocate a tfm (a transformation object) and set the key.
         *
         * In real-world use, a tfm and key are typically used for many
         * encryption/decryption operations.  But in this example, we'll just do a
         * single encryption operation with it (which is not very efficient).
         */

        tfm = crypto_alloc_skcipher("xts(aes)", 0, 0);
        if (IS_ERR(tfm)) {
                pr_err("Error allocating xts(aes) handle: %ld\n", PTR_ERR(tfm));
                return PTR_ERR(tfm);
        }

        get_random_bytes(key, sizeof(key));
        err = crypto_skcipher_setkey(tfm, key, sizeof(key));
        if (err) {
                pr_err("Error setting key: %d\n", err);
                goto out;
        }

        /* Allocate a request object */
        req = skcipher_request_alloc(tfm, GFP_KERNEL);
        if (!req) {
                err = -ENOMEM;
                goto out;
        }

        /* Prepare the input data */
        data = kmalloc(datasize, GFP_KERNEL);
        if (!data) {
                err = -ENOMEM;
                goto out;
        }
        get_random_bytes(data, datasize);

        /* Initialize the IV */
        get_random_bytes(iv, sizeof(iv));

        /*
         * Encrypt the data in-place.
         *
         * For simplicity, in this example we wait for the request to complete
         * before proceeding, even if the underlying implementation is asynchronous.
         *
         * To decrypt instead of encrypt, just change crypto_skcipher_encrypt() to
         * crypto_skcipher_decrypt().
         */
        sg_init_one(&sg, data, datasize);
        skcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG |
                                           CRYPTO_TFM_REQ_MAY_SLEEP,
                                      crypto_req_done, &wait);
        skcipher_request_set_crypt(req, &sg, &sg, datasize, iv);
        err = crypto_wait_req(crypto_skcipher_encrypt(req), &wait);
        if (err) {
                pr_err("Error encrypting data: %d\n", err);
                goto out;
        }
        pr_debug("Encryption was successful\n");
out:
        crypto_free_skcipher(tfm);
        skcipher_request_free(req);
        kfree(data);
        return err;
}
```

Code Example For Use of Operational State Memory With SHASH

```
struct sdesc {
    struct shash_desc shash;
    char ctx[];
};

static struct sdesc *init_sdesc(struct crypto_shash *alg)
{
    struct sdesc *sdesc;
    int size;

    size = sizeof(struct shash_desc) + crypto_shash_descsize(alg);
    sdesc = kmalloc(size, GFP_KERNEL);
    if (!sdesc)
        return ERR_PTR(-ENOMEM);
    sdesc->shash.tfm = alg;
    return sdesc;
}

static int calc_hash(struct crypto_shash *alg,
             const unsigned char *data, unsigned int datalen,
             unsigned char *digest)
{
    struct sdesc *sdesc;
    int ret;

    sdesc = init_sdesc(alg);
    if (IS_ERR(sdesc)) {
        pr_info("can't alloc sdesc\n");
        return PTR_ERR(sdesc);
    }

    ret = crypto_shash_digest(&sdesc->shash, data, datalen, digest);
    kfree(sdesc);
    return ret;
}

static int test_hash(const unsigned char *data, unsigned int datalen,
             unsigned char *digest)
{
    struct crypto_shash *alg;
    char *hash_alg_name = "sha1-padlock-nano";
    int ret;

    alg = crypto_alloc_shash(hash_alg_name, 0, 0);
    if (IS_ERR(alg)) {
            pr_info("can't alloc alg %s\n", hash_alg_name);
            return PTR_ERR(alg);
    }
    ret = calc_hash(alg, data, datalen, digest);
    crypto_free_shash(alg);
    return ret;
}
```

Code Example For Random Number Generator Usage

```
static int get_random_numbers(u8 *buf, unsigned int len)
{
    struct crypto_rng *rng = NULL;
    char *drbg = "drbg_nopr_sha256"; /* Hash DRBG with SHA-256, no PR */
    int ret;

    if (!buf || !len) {
        pr_debug("No output buffer provided\n");
        return -EINVAL;
    }

    rng = crypto_alloc_rng(drbg, 0, 0);
    if (IS_ERR(rng)) {
        pr_debug("could not allocate RNG handle for %s\n", drbg);
        return PTR_ERR(rng);
    }

    ret = crypto_rng_get_bytes(rng, buf, len);
    if (ret < 0)
        pr_debug("generation of random numbers failed\n");
    else if (ret == 0)
        pr_debug("RNG returned no data");
    else
        pr_debug("RNG returned %d bytes of data\n", ret);

out:
    crypto_free_rng(rng);
    return ret;
}
```