NAME

EVP_KDF−SS − The Single Step / One Step EVP_KDF implementation

DESCRIPTION

The EVP_KDF−SS algorithm implements the Single Step key derivation function (SSKDF). SSKDF derives a key using input such as a shared secret key (that was generated during the execution of a key establishment scheme) and fixedinfo. SSKDF is also informally referred to as 'Concat KDF'.

Auxiliary function

The implementation uses a selectable auxiliary function H, which can be one of:

H(x) = hash(x, digest=md)
H(x) = HMAC_hash(x, key=salt, digest=md)
H(x) = KMACxxx(x, key=salt, custom="KDF", outlen=mac_size)

Both the HMAC and KMAC implementations set the key using the 'salt' value. The hash and HMAC also require the digest to be set.

Identity

"SSKDF" is the name for this implementation; it can be used with the **EVP_KDF_fetch()** function.

Supported parameters

The supported parameters are:

"properties" (**OSSL_KDF_PARAM_PROPERTIES**) <UTF8 string>
"digest" (**OSSL_KDF_PARAM_DIGEST**) <UTF8 string>
"mac" (**OSSL_KDF_PARAM_MAC**) <UTF8 string>
"maclen" (**OSSL_KDF_PARAM_MAC_SIZE**) <unsigned integer>
"salt" (**OSSL_KDF_PARAM_SALT**) <octet string>
These parameters work as described in "PARAMETERS" in **EVP_KDF**(3).

"key" (**EVP_KDF_CTRL_SET_KEY**) <octet string>
This parameter set the shared secret that is used for key derivation.

"info" (**OSSL_KDF_PARAM_INFO**) <octet string>
This parameter sets an optional value for fixedinfo, also known as otherinfo.

NOTES

A context for SSKDF can be obtained by calling:

```
EVP_KDF *kdf = EVP_KDF_fetch(NULL, "SSKDF", NULL);
EVP_KDF_CTX *kctx = EVP_KDF_CTX_new(kdf);
```

The output length of an SSKDF is specified via the *keylen* parameter to the **EVP_KDF_derive**(3) function.

EXAMPLES

This example derives 10 bytes using H(x) = SHA−256, with the secret key "secret" and fixedinfo value "label":

```
EVP_KDF *kdf;
EVP_KDF_CTX *kctx;
unsigned char out[10];
OSSL_PARAM params[4], *p = params;

kdf = EVP_KDF_fetch(NULL, "SSKDF", NULL);
kctx = EVP_KDF_CTX_new(kdf);
EVP_KDF_free(kdf);

*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_DIGEST,
                                        SN_sha256, strlen(SN_sha256));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_KEY,
                                         "secret", (size_t)6);
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_INFO,
                                         "label", (size_t)5);
*p = OSSL_PARAM_construct_end();
if (EVP_KDF_derive(kctx, out, sizeof(out), params) <= 0) {
    error("EVP_KDF_derive");
}

EVP_KDF_CTX_free(kctx);
```

This example derives 10 bytes using H(x) = HMAC(SHA−256), with the secret key “secret”, fixedinfo value “label” and salt “salt”:

```
 EVP_KDF *kdf;
 EVP_KDF_CTX *kctx;
 unsigned char out[10];
 OSSL_PARAM params[6], *p = params;

 kdf = EVP_KDF_fetch(NULL, "SSKDF", NULL);
 kctx = EVP_KDF_CTX_new(kdf);
 EVP_KDF_free(kdf);

 *p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_MAC,
                                         SN_hmac, strlen(SN_hmac));
 *p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_DIGEST,
                                         SN_sha256, strlen(SN_sha256));
 *p++ = OSSL_PARAM_construct_octet_string(EVP_KDF_CTRL_SET_KEY,
                                          "secret", (size_t)6);
 *p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_INFO,
                                          "label", (size_t)5);
 *p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_SALT,
                                          "salt", (size_t)4);
 *p = OSSL_PARAM_construct_end();
 if (EVP_KDF_derive(kctx, out, sizeof(out), params) <= 0) {
     error("EVP_KDF_derive");
 }

 EVP_KDF_CTX_free(kctx);
```

This example derives 10 bytes using H(x) = KMAC128(x,salt,outlen), with the secret key “secret” fixedinfo value “label”, salt of “salt” and KMAC outlen of 20:

```
 EVP_KDF *kdf;
 EVP_KDF_CTX *kctx;
 unsigned char out[10];
 OSSL_PARAM params[7], *p = params;

 kdf = EVP_KDF_fetch(NULL, "SSKDF", NULL);
 kctx = EVP_KDF_CTX_new(kdf);
 EVP_KDF_free(kdf);

 *p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_MAC,
                                         SN_kmac128, strlen(SN_kmac128));
 *p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_DIGEST,
                                         SN_sha256, strlen(SN_sha256));
 *p++ = OSSL_PARAM_construct_octet_string(EVP_KDF_CTRL_SET_KEY,
                                          "secret", (size_t)6);
 *p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_INFO,
                                          "label", (size_t)5);
```

```
        *p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_SALT,
                                                  "salt", (size_t)4);
        *p++ = OSSL_PARAM_construct_size_t(OSSL_KDF_PARAM_MAC_SIZE, (size_t)20);
        *p = OSSL_PARAM_construct_end();
        if (EVP_KDF_derive(kctx, out, sizeof(out), params) <= 0) {
            error("EVP_KDF_derive");
        }

        EVP_KDF_CTX_free(kctx);
```

CONFORMING TO

NIST SP800−56Cr1.

SEE ALSO

EVP_KDF(3), **EVP_KDF_CTX_new**(3), **EVP_KDF_CTX_free**(3), **EVP_KDF_CTX_set_params**(3), **EVP_KDF_CTX_get_kdf_size**(3), **EVP_KDF_derive**(3), ‘‘PARAMETERS’’ in **EVP_KDF**(3)

HISTORY

This functionality was added to OpenSSL 3.0.

COPYRIGHT