OpenSSL Provider Documentation

Developer Manual for OpenSSL Provider

Contents

List of Figures

Part I

Introduction

A provider, in OpenSSL terms, is a unit of code that provides one or more implementations for various operations for diverse algorithms that one might want to perform.

An operation is something one wants to do, such as encryption and decryption, key derivation, MAC calculation, signing and verification, etc.

An algorithm is a named method to perform an operation. Very often, the algorithms revolve around cryptographic operations, but may also revolve around other types of operation, such as managing certain types of objects.

For more details click here

Provider is built with different tables, these tables provide information to openssl about what is supported, where to look for to find implementation, what are the parameters needed for each algorithm and how to communicate parameters and data with the external library.

Some of these tables are given below:

- OSSL_DISPATCH: Provides information about function pointers to the implementation of the algorithm.
- OSSL_ALGORITHM: Provides information about dispatch tables(OSSL_DISPATCH) and what is supported by provider.

Part II

Control Flow

To start with, please compile in debug mode with verbose output. To enable verbose output, define DEBUG inside `include/debug.h`. To build in debug mode append `-DCMAKE_BUILD_TYPE=debug` to cmake configuration command

First function to be executed from the provider is `OSSL_provider_init`, this function should be present in the provider with this exact name.

```
OPENSSL_EXPORT
int
OSSL_provider_init(const OSSL_CORE_HANDLE* core,
                   const OSSL_DISPATCH*    in,
                   const OSSL_DISPATCH**   out,
                   void**                  vprovctx)
{
    alc_prov_ctx_p ctx;
    ctx = ALCP_prov_newctx(core, in);

    if (!ctx)
        return 0;

    *out      = ALC_dispatch_table;
    *vprovctx = ctx;

    return 1;
}
```

For more information click here

Be mindfull about how to terminate each table, if table is not terminalted with proper `NULL`, it can lead to a segfault.

Example dispatch table for whole provider

```
static const OSSL_DISPATCH ALC_dispatch_table[] = {
    { OSSL_FUNC_PROVIDER_QUERY_OPERATION, (fptr_t)ALCP_query_operation },
    { OSSL_FUNC_PROVIDER_GET_REASON_STRINGS,
        (fptr_t)ALCP_get_reason_strings },
    { OSSL_FUNC_PROVIDER_GET_PARAMS, (fptr_t)ALCP_get_params },
    { OSSL_FUNC_PROVIDER_TEARDOWN, (fptr_t)ALCP_teardown },
    { 0, NULL }
};
```

Once the `dispatch table` for the entire provider is understood by OpenSSL, OpenSSL calls each function presented in it to understand more of what we actually provide.

```
static const OSSL_ALGORITHM*
ALCP_query_operation(void* vctx, int operation_id, const int* no_cache)
{
    switch (operation_id) {
        case OSSL_OP_DIGEST:
            return ALC_prov_digests;
            break;
        case OSSL_OP_RAND:
            return ALC_prov_rng;
            break;
        default:
            break;
    }

    return NULL;
}
```

This `query_operation` should return another algorithm dispatch table, where we have to specify what dispatch table handles a perticular algorithm.

```
#define ALCP_PROV_NAMES_SHA2_224 \
    "SHA2-224:SHA-224:SHA224:2.16.840.1.101.3.4.2.4"
#define ALCP_PROV_NAMES_SHA2_256 \
    "SHA2-256:SHA-256:SHA256:2.16.840.1.101.3.4.2.1"
#define ALCP_PROV_NAMES_SHA2_384 \
    "SHA2-384:SHA-384:SHA384:2.16.840.1.101.3.4.2.2"
#define ALCP_PROV_NAMES_SHA2_512 \
    "SHA2-512:SHA-512:SHA512:2.16.840.1.101.3.4.2.3"
const OSSL_ALGORITHM ALC_prov_digests[] = {
    { ALCP_PROV_NAMES_SHA2_224, DIGEST_DEF_PROP, sha224_sha2_functions },
    { ALCP_PROV_NAMES_SHA2_256, DIGEST_DEF_PROP, sha256_sha2_functions },
    { ALCP_PROV_NAMES_SHA2_384, DIGEST_DEF_PROP, sha384_sha2_functions },
    { ALCP_PROV_NAMES_SHA2_512, DIGEST_DEF_PROP, sha512_sha2_functions },
    { NULL, NULL, NULL },
};
```

Names of each provider algorithm can be found from names.h from openssl source code.

```
static OSSL_FUNC_digest_newctx_fn ALCP_prov_sha224_sha2_newctx;
static void* ALCP_prov_sha224_sha2_newctx(void* provctx) {
    return ALCP_prov_digest_newctx( provctx, &s_digest_sha224_sha2_ALC_DIGEST_LEN_224_in
}
const OSSL_DISPATCH sha224_sha2_functions[] = {
    { OSSL_FUNC_DIGEST_GET_PARAMS, (fptr_t)ALCP_prov_digest_get_params },
    { OSSL_FUNC_DIGEST_NEWCTX, (fptr_t)ALCP_prov_sha224_sha2_newctx },
    { OSSL_FUNC_DIGEST_DUPCTX, (fptr_t)ALCP_prov_digest_dupctx },
    { OSSL_FUNC_DIGEST_FREECTX, (fptr_t)ALCP_prov_digest_freectx },
    { OSSL_FUNC_DIGEST_GETTABLE_PARAMS, (fptr_t)ALCP_prov_digest_gettable_params },
    { OSSL_FUNC_DIGEST_GETTABLE_CTX_PARAMS, (fptr_t)ALCP_prov_digest_gettable_params },
    { OSSL_FUNC_DIGEST_GET_CTX_PARAMS, (fptr_t)ALCP_prov_sha2_get_ctx_params },
    { OSSL_FUNC_DIGEST_INIT, (fptr_t)ALCP_prov_digest_init },
    { OSSL_FUNC_DIGEST_SETTABLE_CTX_PARAMS, (fptr_t)ALCP_prov_digest_settable_ctx_params
    { OSSL_FUNC_DIGEST_SET_CTX_PARAMS, (fptr_t)ALCP_prov_sha2_set_ctx_params },
    { OSSL_FUNC_DIGEST_UPDATE, (fptr_t)ALCP_prov_digest_update },
    { OSSL_FUNC_DIGEST_FINAL, (fptr_t)ALCP_prov_digest_final },
}
```

Header file for the above OSSL_FUNC params can be found in openssl documentation.

Above dispatch table is constructed by `CREATE_DIGEST_DISPATCHERS` macro in `alcp_digest_prov.h`. I have expanded it manually to give a better understanding.

Once OpenSSL knows SHA2-224 is to be called, it will query the above table, and direct calls are made into the functions to complete the operation.

You can read more about the above digest functions inside openssl's own documentation.

Functions has a particular declaration which needs to be followed. OpenSSL has already defined macros which help us declare the functions in the correct manner. Please click here to see the macro.

Example of function defined with the macro.

```
OSSL_FUNC_digest_dupctx_fn         ALCP_prov_digest_dupctx;
OSSL_FUNC_digest_freectx_fn        ALCP_prov_digest_freectx;
OSSL_FUNC_digest_get_ctx_params_fn ALCP_prov_digest_get_ctx_params;
OSSL_FUNC_digest_set_ctx_params_fn ALCP_prov_digest_set_ctx_params;
OSSL_FUNC_digest_update_fn         ALCP_prov_digest_update;
OSSL_FUNC_digest_final_fn          ALCP_prov_digest_final;
OSSL_FUNC_digest_get_params_fn     ALCP_prov_digest_get_params;
```