Intel Intelligent Storage Acceleration Library Crypto

2.25.0

Generated by Doxygen 1.9.8

Chapter 1

Intel(R) Intelligent Storage Acceleration Library Crypto Version

ISA-L_crypto is a collection of optimized low-level functions targeting storage applications. ISA-L_crypto includes:

- Multi-buffer hashes - run multiple hash jobs together on one core for much better throughput than single-buffer versions.
  - SHA1, SHA256, SHA512, MD5, SM3
- Multi-hash - Get the performance of multi-buffer hashing with a single-buffer interface. Specification ref : `Multi-Hash white paper`
- Multi-hash + murmur - run both together.
- AES - block ciphers
  - XTS, GCM, CBC
- Rolling hash - Hash input in a window which moves through the input

Also see:

- `ISA-L_crypto for updates`.
- For non-crypto ISA-L see `isa-l on github`.
- The `github wiki` covering isa-l and isa-l crypto.
- Contributing.
- Security Policy.
- FIPS Mode.

1.1 Building ISA-L

1.1.1 Prerequisites

x86_64:

- Assembler: nasm v2.14.01 or later
- Compiler: gcc, clang, icc or MSVC (Visual Studio 2019 or later).
- Make: GNU 'make' or 'nmake' (Windows).
- Optional: Building with autotools requires autoconf/automake packages.

aarch64:

- Assembler: gas v2.34 or later.
- Compiler: gcc v8 or later.
- For gas v2.24∼v2.34, sve2 instructions are not supported. To workaround it, sve2 optimization should be disabled by
  - ./configure –disable-sve2
  - make -f Makefile.unx DEFINES+=-DNO_SVE2=1

1.1.2 Autotools

To build and install the library with autotools it is usually sufficient to run:

```
./autogen.sh
./configure
make
sudo make install
```

1.1.3 Makefile

To use a standard makefile run:

```
make -f Makefile.unx
```

1.1.4 Windows

On Windows use nmake to build dll and static lib:

```
nmake -f Makefile.nmake
```

1.1.5 Other make targets

Other targets include:

- `make check` : create and run tests
- `make tests` : create additional unit tests
- `make perfs` : create included performance tests
- `make ex` : build examples
- `make doc` : build API manual

1.2 Algorithm recommendations

Legacy or to be avoided algorithms listed in the table below are implemented in the library in order to support legacy applications. Please use corresponding alternative algorithms instead.

#	Algorithm	Recommendation	Alternative
1	MD5 integrity	Legacy	SHA256
2	SHA1 integrity	Avoid	SHA256

Intel(R) Intelligent Storage Acceleration for Crypto Library depends on C library and it is recommended to use its latest version.

Applications using the Intel(R) Intelligent Storage Acceleration for Crypto Library rely on Operating System to provide process isolation. As the result, it is recommended to use latest Operating System patches and security updates.

1.3 DLL Injection Attack

1.3.1 Problem

The Windows OS has an insecure predefined search order and set of defaults when trying to locate a resource. If the resource location is not specified by the software, an attacker need only place a malicious version in one of the locations Windows will search, and it will be loaded instead. Although this weakness can occur with any resource, it is especially common with DLL files.

1.3.2 Solutions

Applications using libisal_crypto DLL library may need to apply one of the solutions to prevent from DLL injection attack.

Two solutions are available:

- Using a Fully Qualified Path is the most secure way to load a DLL
- Signature verification of the DLL

1.3.3 Resources and Solution Details

- Security remarks section of LoadLibraryEx documentation by Microsoft: https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibraryexa#security-remark
- Microsoft Dynamic Link Library Security article: https://docs.microsoft.com/en-us/windows/win32/dlls/d
- Hijack Execution Flow: DLL Search Order Hijacking: https://attack.mitre.org/techniques/T1574/001
- Hijack Execution Flow: DLL Side-Loading: https://attack.mitre.org/techniques/T1574/002

Chapter 2

Contributing to ISA-L_crypto

Everyone is welcome to contribute. Patches may be submitted using GitHub pull requests (PRs). All commits must be signed off by the developer (–signoff) which indicates that you agree to the Developer Certificate of Origin. Patch discussion will happen directly on the GitHub PR. Design pre-work and general discussion occurs on the mailing list. Anyone can provide feedback in either location and all discussion is welcome. Decisions on whether to merge patches will be handled by the maintainer.

2.1 License

ISA-L_crypto is licensed using a BSD 3-clause [license]. All code submitted to the project is required to carry that license.

2.2 Certificate of Origin

In order to get a clear contribution chain of trust we use the signed-off-by language used by the Linux kernel project.

2.3 Mailing List

Contributors and users are welcome to submit new request on our roadmap, submit patches, file issues, and ask questions on our mailing list.

2.4 Coding Style

The coding style for ISA-L_crypto C code is roughly based on LLVM style with some customizations. Use the included format script to format C code.

```
./tools/format.sh
```

And use check format script before submitting.

```
./tools/check_format.sh
```

Chapter 3

ISA-L Security Policy

3.1 Report a Vulnerability

Please report security issues or vulnerabilities to the Intel Security Center.

For more information on how Intel works to resolve security issues, see Vulnerability Handling Guidelines.

3.2 Security Considerations & Options for Increased Security

3.2.1 Security Considerations

The security of a system that uses cryptography depends on the strength of the cryptographic algorithms as well as the strength of the keys. Cryptographic key strength is dependent on several factors, with some of the most important factors including the length of the key, the entropy of the key bits, and maintaining the secrecy of the key.

The selection of an appropriate algorithm and mode of operation critically affects the security of a system. Appropriate selection criteria is beyond the scope of this document and should be determined based upon usage, appropriate standards and consultation with a cryptographic expert. This library includes some algorithms, which are considered cryptographically weak and are included only for legacy and interoperability reasons. See the "Recommendations" section for more details.

Secure creation of key material is not a part of this library. This library assumes that cryptographic keys have been created using approved methods with an appropriate and secure entropy source. Users of this library are referred to NIST SP800-133 Revision 1, Recommendation for Cryptographic Key Generation, found at https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r1.pdf

Even with the use of strong cryptographic algorithms and robustly generated keys, software implementations of cryptographic algorithms may be attacked at the implementation through cache-timing attacks, buffer-over-reads, and other software vulnerabilities. Counter-measures against these types of attacks are possible but require additional processing cycles. Whether a particular system should provide such counter-measures depends on the threats to that system, and cannot be determined by a general library such as this one. In order to provide the most flexible implementation, this library allows certain counter-measures to be enabled or disabled at compile time. These options are listed below as the "Options for Increased Security" and are enabled through various build flags.

3.2.2 Options for Increased Security

There are two build options that are used to increase safety in the code and help protect external functions from incorrect input data. The SAFE_DATA and SAFE_PARAM options are enabled by default. Due to the potential performance impact associated to the extra code, these can be disabled by setting the parameter equal to "n" (e.g. make -f Makefile.unx SAFE_PARAM=n).

No specific code has been added, and no specific validation or security tests have been performed to help protect against or check for side-channel attacks.

3.2.3 SAFE_DATA

Stack and registers containing sensitive information, such as keys, are cleared upon completion of a function call.

3.2.4 SAFE_PARAM

Input parameters are checked, looking generally for NULL pointers or an incorrect input length.

3.2.5 Galois Counter Mode (GCM) TAG Size

The library GCM implementation provides flexibility as to tag size selection. As explained in NIST Special Publication 800-38D section 5.2.1.2 and Appendix C, using tag sizes shorter than 96 bits can be insecure. Please refer to the aforementioned sections to understand the details, trade offs and mitigations of using shorter tag sizes.

Chapter 4

FIPS Mode on ISA-L Crypto

4.1 Compilation

FIPS mode is disabled in the library by default. In order to enable it, the library needs to be compiled as follows:

- Using autotools:

```
./autogen.sh
./configure --enable-fips-mode
make
```

- Standard makefile:

```
make -f Makefile.unx FIPS_MODE=y
```

- Windows Makefile:

```
make /f Makefile.nmake FIPS_MODE=y
```

4.2 Covered API by this mode

Only the "isal_" prefixed API is in the scope of this mode (e.g. `isal_aes_cbc_enc_128()`).

isal_crypto.h or isal_crypto_api.h must be included in the application/framework calling this API.

After the first call on this API, crypto self tests will be run. If any of the tests fail, no crypto operation will be performed and the API will return ISAL_CRYPTO_ERR_SELF_TEST. Subsequent calls will return this error too.

The self tests can also be run at the application level by calling explicitly `isal_self_tests()`.

The validation of self tests is executed only once, either by invoking the `isal_self_tests()` function or by invoking a covered crypto function, such as `isal_aes_cbc_enc_128()`. After the tests have been run once, they will not be executed again, and subsequent API calls will use the previous test result.

If an algorithm is not NIST approved (e.g. SM3), calling the crypto function will return ISAL_CRYPTO_ERR_FIPS_↩ INVALID_ALGO.

4.3 Example of usage

```
#include <isal_crypto_api.h>
#include <aes_cbc.h>

...

int ret = isal_aes_cbc_enc_128(pt, iv, expkey_enc, ct, pt_len);
if (ret != 0)
        exit(1);
```

4.4 Considerations

- This library does not check for uniqueness on AES-GCM key/IV pair.
- FIPS mode is supported from ISA-L Crypto version v2.25.
- FIPS mode has only been tested on Intel x86 architecture.

Chapter 5

Deprecated List

Global aes_cbc_dec_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_dec_128() instead.

Global aes_cbc_dec_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_dec_192() instead.

Global aes_cbc_dec_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_dec_256() instead.

Global aes_cbc_enc_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_enc_128() instead.

Global aes_cbc_enc_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_enc_192() instead.

Global aes_cbc_enc_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

Please use isal_aes_cbc_enc_256() instead.

Global aes_cbc_precomp (uint8_t *key, int key_size, struct isal_cbc_key_data *keys_blk)

Please use isal_aes_keyexp_128(), isal_aes_keyexp_192() or isal_aes_keyexp_256() instead.

Global aes_gcm_dec_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_128() instead.

Global aes_gcm_dec_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_128_finalize() instead.

Global aes_gcm_dec_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_128_nt() instead.

Global aes_gcm_dec_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_dec_128_update() instead.

Global aes_gcm_dec_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_dec_128_update_nt() instead.

Global aes_gcm_dec_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_256() instead.

Global aes_gcm_dec_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_256_finalize() instead.

Global aes_gcm_dec_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_dec_256_nt() instead.

Global aes_gcm_dec_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_dec_256_update() instead.

Global aes_gcm_dec_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_dec_256_update_nt() instead.

Global aes_gcm_enc_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_128() instead.

Global aes_gcm_enc_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_128_finalize() instead.

Global aes_gcm_enc_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_128_nt() instead.

Global aes_gcm_enc_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_enc_128_update() instead.

Global aes_gcm_enc_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_enc_128_update_nt() instead.

Global aes_gcm_enc_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_256() instead.

Global aes_gcm_enc_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_256_finalize() instead.

Global aes_gcm_enc_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

Please use isal_aes_gcm_enc_256_nt() instead.

Global aes_gcm_enc_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_enc_256_update() instead.

Global aes_gcm_enc_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Please use isal_aes_gcm_enc_256_update_nt() instead.

Global aes_gcm_init_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

Please use isal_aes_gcm_init_128() instead.

Global aes_gcm_init_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

Please use isal_aes_gcm_init_256() instead.

Global aes_gcm_pre_128 (const void *key, struct isal_gcm_key_data *key_data)

Please use isal_aes_gcm_pre_128() instead.

Global aes_gcm_pre_256 (const void *key, struct isal_gcm_key_data *key_data)

Please use isal_aes_gcm_pre_256() instead.

Global aes_keyexp_128 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

Please use isal_aes_keyexp_128() instead.

Global aes_keyexp_192 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

Please use isal_aes_keyexp_192() instead.

Global aes_keyexp_256 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

Please use isal_aes_keyexp_256() instead.

Global md5_ctx_mgr_flush (ISAL_MD5_HASH_CTX_MGR *mgr)

Please use isal_md5_ctx_mgr_submit() instead.

Global md5_ctx_mgr_init (ISAL_MD5_HASH_CTX_MGR *mgr)

Please use isal_md5_ctx_mgr_init() instead.

Global md5_ctx_mgr_submit (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

Please use isal_md5_ctx_mgr_submit() instead.

Global mh_sha1_finalize (struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest)

Please use isal_mh_sha1_finalize() instead.

Global mh_sha1_finalize_base (struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest)

Please use isal_mh_sha1_finalize() instead.

Global mh_sha1_init (struct isal_mh_sha1_ctx *ctx)

Please use isal_mh_sha1_init() instead.

Global mh_sha1_murmur3_x64_128_finalize (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_sha1_digest, void *murmur3_x64_128_digest)

Please use isal_mh_sha1_murmur3_x64_128_finalize() instead.

Global mh_sha1_murmur3_x64_128_finalize_base (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_sha1_digest, void *murmur3_x64_128_digest)

Please use isal_mh_sha1_murmur3_x64_128_finalize() instead.

Global mh_sha1_murmur3_x64_128_init (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, uint64_t murmur_seed)

Please use isal_mh_sha1_murmur3_x64_128_init() instead.

Global mh_sha1_murmur3_x64_128_update (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha1_murmur3_x64_128_update() instead.

Global mh_sha1_murmur3_x64_128_update_base (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha1_murmur3_x64_128_update() instead.

Global mh_sha1_update (struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha1_update() instead.

Global mh_sha1_update_base (struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha1_update() instead.

Global mh_sha256_finalize (struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest)

Please use isal_mh_sha256_finalize() instead.

Global mh_sha256_finalize_base (struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest)

Please use isal_mh_sha256_finalize() instead.

Global mh_sha256_init (struct isal_mh_sha256_ctx *ctx)

Please use isal_mh_sha256_init() instead.

Global mh_sha256_update (struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha256_update() instead.

Global mh_sha256_update_base (struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len)

Please use isal_mh_sha256_update() instead.

Global rolling_hash2_init (struct isal_rh_state2 *state, uint32_t w)

Please use isal_rolling_hash2_init() instead.

Global rolling_hash2_reset (struct isal_rh_state2 *state, uint8_t *init_bytes)

Please use isal_rolling_hash2_reset() instead.

Global rolling_hash2_run (struct isal_rh_state2 *state, uint8_t *buffer, uint32_t max_len, uint32_t mask, uint32_t trigger, uint32_t *offset)

Please use isal_rolling_hash2_run() instead.

Global rolling_hashx_mask_gen (long mean, int shift)

Please use isal_rolling_hashx_mask_gen() instead.

Global sha1_ctx_mgr_flush (ISAL_SHA1_HASH_CTX_MGR *mgr)

Please use isal_sha1_ctx_mgr_flush() instead.

Global sha1_ctx_mgr_init (ISAL_SHA1_HASH_CTX_MGR *mgr)

Please use isal_sha1_ctx_mgr_init() instead.

Global sha1_ctx_mgr_submit (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

Please use isal_sha1_ctx_mgr_submit() instead.

Global sha256_ctx_mgr_flush (ISAL_SHA256_HASH_CTX_MGR *mgr)

Please use isal_sha256_ctx_mgr_flush() instead.

Global sha256_ctx_mgr_init (ISAL_SHA256_HASH_CTX_MGR *mgr)

Please use isal_sha256_ctx_mgr_init() instead.

Global sha256_ctx_mgr_submit (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

Please use isal_sha256_ctx_mgr_submit() instead.

Global sha512_ctx_mgr_flush (ISAL_SHA512_HASH_CTX_MGR *mgr)

Please use isal_sha512_ctx_mgr_flush() instead.

Global sha512_ctx_mgr_init (ISAL_SHA512_HASH_CTX_MGR *mgr)

Please use isal_sha512_ctx_mgr_init() instead.

Global sha512_ctx_mgr_submit (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

Please use isal_sha512_ctx_mgr_submit() instead.

Global sm3_ctx_mgr_flush (ISAL_SM3_HASH_CTX_MGR *mgr)

Please use isal_sm3_ctx_mgr_flush() instead.

Global sm3_ctx_mgr_init (ISAL_SM3_HASH_CTX_MGR *mgr)

Please use isal_sm3_ctx_mgr_init() instead.

Global sm3_ctx_mgr_submit (ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

Please use isal_sm3_ctx_mgr_submit() instead.

Global XTS_AES_128_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

Please use isal_aes_xts_dec_128() instead.

Global XTS_AES_128_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

Please use isal_aes_xts_dec_128_expanded_key() instead.

Global XTS_AES_128_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

Please use isal_aes_xts_enc_128() instead.

Global XTS_AES_128_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

Please use isal_aes_xts_enc_128_expanded_key() instead.

Global XTS_AES_256_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

Please use isal_aes_xts_dec_256() instead.

Global XTS_AES_256_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

Please use isal_aes_xts_dec_256_expanded_key() instead.

Global XTS_AES_256_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

Please use isal_aes_xts_enc_256() instead.

Global XTS_AES_256_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

Please use isal_aes_xts_enc_256_expanded_key() instead.

Chapter 6

Instruction Set Requirements for arch-specific functions (non-multibinary)

Global aes_cbc_dec_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_dec_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_dec_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_enc_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_enc_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_enc_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)

SSE4.1 and AESNI

Global aes_cbc_precomp (uint8_t *key, int key_size, struct isal_cbc_key_data *keys_blk)

SSE4.1 and AESNI

Global aes_gcm_dec_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_dec_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_dec_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_dec_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_dec_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_enc_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_enc_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_enc_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

SSE4.1 and AESNI

Global aes_gcm_enc_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_enc_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

SSE4.1 and AESNI

Global aes_gcm_init_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

SSE4.1 and AESNI

Global aes_gcm_init_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

SSE4.1 and AESNI

Global aes_gcm_pre_128 (const void *key, struct isal_gcm_key_data *key_data)

SSE4.1 and AESNI

Global aes_gcm_pre_256 (const void *key, struct isal_gcm_key_data *key_data)

SSE4.1 and AESNI

Global aes_keyexp_128 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

SSE4.1

Global aes_keyexp_192 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

SSE4.1

Global aes_keyexp_256 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

SSE4.1

Global isal_aes_cbc_dec_128 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_cbc_dec_192 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_cbc_dec_256 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_cbc_enc_128 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_cbc_enc_192 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_cbc_enc_256 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_dec_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_enc_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_init_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_init_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_pre_128 (const void *key, struct isal_gcm_key_data *key_data)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_gcm_pre_256 (const void *key, struct isal_gcm_key_data *key_data)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_keyexp_128 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_keyexp_192 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_keyexp_256 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_dec_128 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_dec_128_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_dec_256 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_dec_256_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_enc_128 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_enc_128_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_enc_256 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_aes_xts_enc_256_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

AES extensions and SSE4.1 for x86 or ASIMD for ARM

Global isal_md5_ctx_mgr_flush (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX **ctx_out)

SSE4.1 for x86 or ASIMD for ARM

Global isal_md5_ctx_mgr_init (ISAL_MD5_HASH_CTX_MGR *mgr)

SSE4.1 for x86 or ASIMD for ARM

Global isal_md5_ctx_mgr_submit (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx_in, ISAL_MD5_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha1_ctx_mgr_flush (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX **ctx_out)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha1_ctx_mgr_init (ISAL_SHA1_HASH_CTX_MGR *mgr)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha1_ctx_mgr_submit (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx_in, ISAL_SHA1_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha256_ctx_mgr_flush (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX **ctx↩ _out)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha256_ctx_mgr_init (ISAL_SHA256_HASH_CTX_MGR *mgr)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha256_ctx_mgr_submit (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx↩ _in, ISAL_SHA256_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX↩ _FLAG flags)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha512_ctx_mgr_flush (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX **ctx↩ _out)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha512_ctx_mgr_init (ISAL_SHA512_HASH_CTX_MGR *mgr)

SSE4.1 for x86 or ASIMD for ARM

Global isal_sha512_ctx_mgr_submit (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx↩ _in, ISAL_SHA512_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX↩ _FLAG flags)

SSE4.1 for x86 or ASIMD for ARM

Global md5_ctx_mgr_flush (ISAL_MD5_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global md5_ctx_mgr_init (ISAL_MD5_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global md5_ctx_mgr_submit (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

SSE4.1 or AVX or AVX2 or AVX512

Global sha1_ctx_mgr_flush (ISAL_SHA1_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global sha1_ctx_mgr_init (ISAL_SHA1_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global sha1_ctx_mgr_submit (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

SSE4.1 or AVX or AVX2 or AVX512

Global sha256_ctx_mgr_flush (ISAL_SHA256_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2

Global sha256_ctx_mgr_init (ISAL_SHA256_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2

Global sha256_ctx_mgr_submit (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

SSE4.1 or AVX or AVX2

Global sha512_ctx_mgr_flush (ISAL_SHA512_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global sha512_ctx_mgr_init (ISAL_SHA512_HASH_CTX_MGR *mgr)

SSE4.1 or AVX or AVX2 or AVX512

Global sha512_ctx_mgr_submit (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

SSE4.1 or AVX or AVX2 or AVX512

Global XTS_AES_128_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

AES-NI

Global XTS_AES_128_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

AES-NI

Global XTS_AES_128_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

AES-NI

Global XTS_AES_128_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

AES-NI

Global XTS_AES_256_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

AES-NI

Global XTS_AES_256_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

AES-NI

Global XTS_AES_256_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

AES-NI

Global XTS_AES_256_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

AES-NI

Chapter 7

Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 9

Data Structure Documentation

9.1 isal_cbc_key_data Struct Reference

holds intermediate key data used in encryption/decryption

`#include <aes_cbc.h>`

9.1.1 Detailed Description

holds intermediate key data used in encryption/decryption

The documentation for this struct was generated from the following file:

- aes_cbc.h

9.2 isal_gcm_context_data Struct Reference

holds GCM operation context

`#include <aes_gcm.h>`

9.2.1 Detailed Description

holds GCM operation context

The documentation for this struct was generated from the following file:

- aes_gcm.h

9.3 isal_gcm_key_data Struct Reference

holds intermediate key data needed to improve performance

`#include <aes_gcm.h>`

9.3.1 Detailed Description

holds intermediate key data needed to improve performance

isal_gcm_key_data hold internal key information used by gcm128 and gcm256.

The documentation for this struct was generated from the following file:

- aes_gcm.h

9.4 ISAL_MD5_HASH_CTX Struct Reference

Context layer - Holds info describing a single MD5 job for the multi-buffer CTX manager.

`#include <md5_mb.h>`

Collaboration diagram for ISAL_MD5_HASH_CTX:

9.5 ISAL_MD5_HASH_CTX_MGR Struct Reference

Context layer - Holds state for multi-buffer MD5 jobs.

`#include <md5_mb.h>`

Collaboration diagram for ISAL_MD5_HASH_CTX_MGR:

9.5.1 Detailed Description

Context layer - Holds state for multi-buffer MD5 jobs.

The documentation for this struct was generated from the following file:

- md5_mb.h

9.6 ISAL_MD5_JOB Struct Reference

Scheduler layer - Holds info describing a single MD5 job for the multi-buffer manager.

`#include <md5_mb.h>`

Data Fields

- uint8_t * **buffer**
  pointer to data buffer for this job
- uint32_t **len**
  length of buffer for this job in blocks.
- ALIGN uint32_t **result_digest** [ISAL_MD5_DIGEST_NWORDS]
  Digest output (in array of little-endian double words, different than SHA's).
- ISAL_JOB_STS **status**
  output job status
- void * **user_data**
  pointer for user's job-related data

9.6.1 Detailed Description

Scheduler layer - Holds info describing a single MD5 job for the multi-buffer manager.

The documentation for this struct was generated from the following file:

- md5_mb.h

9.7 ISAL_MD5_LANE_DATA Struct Reference

Scheduler layer - Lane data.

`#include <md5_mb.h>`

Collaboration diagram for ISAL_MD5_LANE_DATA:

9.7.1 Detailed Description

Scheduler layer - Lane data.

The documentation for this struct was generated from the following file:

- md5_mb.h

9.8 ISAL_MD5_MB_ARGS_X32 Struct Reference

Scheduler layer - Holds arguments for submitted MD5 job.

`#include <md5_mb.h>`

9.8.1 Detailed Description

Scheduler layer - Holds arguments for submitted MD5 job.

The documentation for this struct was generated from the following file:

- md5_mb.h

9.9 ISAL_MD5_MB_JOB_MGR Struct Reference

Scheduler layer - Holds state for multi-buffer MD5 jobs.

`#include <md5_mb.h>`

Collaboration diagram for ISAL_MD5_MB_JOB_MGR:

Data Fields

- uint64_t **unused_lanes** [4]
  each byte or nibble is index (0...31 or 15) of unused lanes.

9.9.1 Detailed Description

Scheduler layer - Holds state for multi-buffer MD5 jobs.

The documentation for this struct was generated from the following file:

- md5_mb.h

9.10 isal_mh_sha1_ctx Struct Reference

Holds info describing a single mh_sha1.

`#include <mh_sha1.h>`

Data Fields

- uint32_t **mh_sha1_digest** [ISAL_SHA1_DIGEST_WORDS]

  the digest of multi-hash SHA1
- uint64_t **total_length**

  Parameters for update feature, describe the lengths of input buffers in bytes.
- uint8_t **partial_block_buffer** [ISAL_MH_SHA1_BLOCK_SIZE ∗2]

  Padding the tail of input data for SHA1.
- uint8_t mh_sha1_interim_digests [sizeof(uint32_t) ∗ISAL_SHA1_DIGEST_WORDS ∗ISAL_HASH_SEGS]
- uint8_t frame_buffer [ISAL_MH_SHA1_BLOCK_SIZE+ISAL_AVX512_ALIGNED]

9.10.1 Detailed Description

Holds info describing a single mh_sha1.

It is better to use heap to allocate this data structure to avoid stack overflow.

9.10.2 Field Documentation

9.10.2.1 frame_buffer

```
uint8_t isal_mh_sha1_ctx::frame_buffer[ISAL_MH_SHA1_BLOCK_SIZE+ISAL_AVX512_ALIGNED]
```

Re-structure sha1 block data from different segments to fit big endian. Use ISAL_AVX512_ALIGNED for 64-byte alignment purpose.

9.10.2.2 mh_sha1_interim_digests

```
uint8_t isal_mh_sha1_ctx::mh_sha1_interim_digests[sizeof(uint32_t) *ISAL_SHA1_DIGEST_WORDS *ISAL↩
_HASH_SEGS]
```

Storing the SHA1 interim digests of all 16 segments. Each time, it will be copied to stack for 64-byte alignment purpose.

The documentation for this struct was generated from the following file:

- mh_sha1.h

9.11 isal_mh_sha1_murmur3_x64_128_ctx Struct Reference

Holds info describing a single mh_sha1_murmur3_x64_128.

```
#include <mh_sha1_murmur3_x64_128.h>
```

Data Fields

- uint32_t **mh_sha1_digest** [ISAL_SHA1_DIGEST_WORDS]

  the digest of multi-hash SHA1
- uint32_t murmur3_x64_128_digest [ISAL_MURMUR3_x64_128_DIGEST_WORDS]
- uint64_t **total_length**

  Parameters for update feature, describe the lengths of input buffers in bytes.
- uint8_t **partial_block_buffer** [ISAL_MH_SHA1_BLOCK_SIZE ∗2]

  Padding the tail of input data for SHA1.
- uint8_t mh_sha1_interim_digests [sizeof(uint32_t) ∗ISAL_SHA1_DIGEST_WORDS ∗ISAL_HASH_SEGS]
- uint8_t frame_buffer [ISAL_MH_SHA1_BLOCK_SIZE+ISAL_AVX512_ALIGNED]

9.11.1 Detailed Description

Holds info describing a single mh_sha1_murmur3_x64_128.

It is better to use heap to allocate this data structure to avoid stack overflow.

9.11.2 Field Documentation

9.11.2.1 frame_buffer

```
uint8_t isal_mh_sha1_murmur3_x64_128_ctx::frame_buffer[ISAL_MH_SHA1_BLOCK_SIZE+ISAL_AVX512_↩
ALIGNED]
```

Re-structure sha1 block data from different segments to fit big endian. Use ISAL_AVX512_ALIGNED for 64-byte alignment purpose.

9.11.2.2 mh_sha1_interim_digests

```
uint8_t isal_mh_sha1_murmur3_x64_128_ctx::mh_sha1_interim_digests[sizeof(uint32_t) *ISAL_SHA1_↩
DIGEST_WORDS *ISAL_HASH_SEGS]
```

Storing the SHA1 interim digests of all 16 segments. Each time, it will be copied to stack for 64-byte alignment purpose.

9.11.2.3 murmur3_x64_128_digest

```
uint32_t isal_mh_sha1_murmur3_x64_128_ctx::murmur3_x64_128_digest[ISAL_MURMUR3_x64_128_DIGEST_↩
WORDS]
```

the digest of murmur3_x64_128

The documentation for this struct was generated from the following file:

- mh_sha1_murmur3_x64_128.h

9.12 isal_mh_sha256_ctx Struct Reference

Holds info describing a single mh_sha256.

`#include <mh_sha256.h>`

Data Fields

- uint32_t **mh_sha256_digest** [ISAL_SHA256_DIGEST_WORDS]

  the digest of multi-hash SHA256
- uint64_t **total_length**

  Parameters for update feature, describe the lengths of input buffers in bytes.
- uint8_t **partial_block_buffer** [ISAL_MH_SHA256_BLOCK_SIZE ∗2]

  Padding the tail of input data for SHA256.
- uint8_t mh_sha256_interim_digests [sizeof(uint32_t) ∗ISAL_SHA256_DIGEST_WORDS ∗ISAL_HASH_SEGS]
- uint8_t frame_buffer [ISAL_MH_SHA256_BLOCK_SIZE+ISAL_AVX512_ALIGNED]

9.12.1 Detailed Description

Holds info describing a single mh_sha256.

It is better to use heap to allocate this data structure to avoid stack overflow.

9.12.2 Field Documentation

9.12.2.1 frame_buffer

`uint8_t isal_mh_sha256_ctx::frame_buffer[ISAL_MH_SHA256_BLOCK_SIZE+ISAL_AVX512_ALIGNED]`

Re-structure sha256 block data from different segments to fit big endian. Use ISAL_AVX512_ALIGNED for 64-byte alignment purpose.

9.12.2.2 mh_sha256_interim_digests

`uint8_t isal_mh_sha256_ctx::mh_sha256_interim_digests[sizeof(uint32_t) *ISAL_SHA256_DIGEST_WORDS * ISAL_HASH_SEGS]`

Storing the SHA256 interim digests of all 16 segments. Each time, it will be copied to stack for 64-byte alignment purpose.

The documentation for this struct was generated from the following file:

- mh_sha256.h

9.13 isal_rh_state2 Struct Reference

Context for rolling_hash2 functions.

`#include <rolling_hashx.h>`

9.13.1 Detailed Description

Context for rolling_hash2 functions.

The documentation for this struct was generated from the following file:

- rolling_hashx.h

9.14 ISAL_SHA1_HASH_CTX Struct Reference

Context layer - Holds info describing a single SHA1 job for the multi-buffer CTX manager. This structure must be allocated to 16-byte aligned memory.

`#include <sha1_mb.h>`

Collaboration diagram for ISAL_SHA1_HASH_CTX:

Data Fields

- ISAL_HASH_CTX_STS **status**
  Context status flag.
- ISAL_HASH_CTX_ERROR **error**
  Context error flag.
- uint64_t **total_length**
  Running counter of length processed for this CTX's job.
- const void * **incoming_buffer**
  pointer to data input buffer for this CTX's job
- uint32_t **incoming_buffer_length**
  length of buffer for this job in bytes.
- uint8_t **partial_block_buffer** [ISAL_SHA1_BLOCK_SIZE *2]
  CTX partial blocks.
- void * **user_data**
  pointer for user to keep any job-related data

9.14.1 Detailed Description

Context layer - Holds info describing a single SHA1 job for the multi-buffer CTX manager. This structure must be allocated to 16-byte aligned memory.

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.15 ISAL_SHA1_HASH_CTX_MGR Struct Reference

Context layer - Holds state for multi-buffer SHA1 jobs.

`#include <sha1_mb.h>`

Collaboration diagram for ISAL_SHA1_HASH_CTX_MGR:

9.15.1 Detailed Description

Context layer - Holds state for multi-buffer SHA1 jobs.

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.16 ISAL_SHA1_JOB Struct Reference

Scheduler layer - Holds info describing a single SHA1 job for the multi-buffer manager.

`#include <sha1_mb.h>`

Data Fields

- uint8_t * **buffer**
  pointer to data buffer for this job
- uint32_t **len**
  length of buffer for this job in blocks.
- ISAL_JOB_STS **status**
  output job status
- void * **user_data**
  pointer for user's job-related data

9.16.1 Detailed Description

Scheduler layer - Holds info describing a single SHA1 job for the multi-buffer manager.

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.17 ISAL_SHA1_LANE_DATA Struct Reference

Scheduler layer - Lane data.

```
#include <sha1_mb.h>
```

Collaboration diagram for ISAL_SHA1_LANE_DATA:

9.17.1 Detailed Description

Scheduler layer - Lane data.

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.18 ISAL_SHA1_MB_ARGS_X16 Struct Reference

Scheduler layer - Holds arguments for submitted SHA1 job.

```
#include <sha1_mb.h>
```

9.18.1 Detailed Description

Scheduler layer - Holds arguments for submitted SHA1 job.

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.19 ISAL_SHA1_MB_JOB_MGR Struct Reference

Scheduler layer - Holds state for multi-buffer SHA1 jobs.

`#include <sha1_mb.h>`

Collaboration diagram for ISAL_SHA1_MB_JOB_MGR:

Data Fields

- uint64_t unused_lanes

9.19.1 Detailed Description

Scheduler layer - Holds state for multi-buffer SHA1 jobs.

9.19.2 Field Documentation

9.19.2.1 unused_lanes

`uint64_t ISAL_SHA1_MB_JOB_MGR::unused_lanes`

each nibble is index (0...3 or 0...7 or 0...15) of unused lanes, nibble 4 or 8 is set to F as a flag

The documentation for this struct was generated from the following file:

- sha1_mb.h

9.20 ISAL_SHA256_HASH_CTX Struct Reference

Context layer - Holds info describing a single SHA256 job for the multi-buffer CTX manager This structure must be allocated to 16-byte aligned memory.

`#include <sha256_mb.h>`

Collaboration diagram for ISAL_SHA256_HASH_CTX:

Data Fields

- ISAL_HASH_CTX_STS **status**

  Context status flag.
- ISAL_HASH_CTX_ERROR **error**

  Context error flag.
- uint64_t **total_length**

  Running counter of length processed for this CTX's job.
- const void * **incoming_buffer**

  pointer to data input buffer for this CTX's job
- uint32_t **incoming_buffer_length**

  length of buffer for this job in bytes.
- uint8_t **partial_block_buffer** [ISAL_SHA256_BLOCK_SIZE *2]

  CTX partial blocks.
- void * **user_data**

  pointer for user to keep any job-related data

9.20.1 Detailed Description

Context layer - Holds info describing a single SHA256 job for the multi-buffer CTX manager This structure must be allocated to 16-byte aligned memory.

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.21 ISAL_SHA256_HASH_CTX_MGR Struct Reference

Context layer - Holds state for multi-buffer SHA256 jobs.

```
#include <sha256_mb.h>
```

Collaboration diagram for ISAL_SHA256_HASH_CTX_MGR:

9.21.1 Detailed Description

Context layer - Holds state for multi-buffer SHA256 jobs.

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.22 ISAL_SHA256_JOB Struct Reference

Scheduler layer - Holds info describing a single SHA256 job for the multi-buffer manager.

`#include <sha256_mb.h>`

Data Fields

- uint8_t * **buffer**
  pointer to data buffer for this job
- uint64_t **len**
  length of buffer for this job in blocks.
- ISAL_JOB_STS **status**
  output job status
- void * **user_data**
  pointer for user's job-related data

9.22.1 Detailed Description

Scheduler layer - Holds info describing a single SHA256 job for the multi-buffer manager.

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.23 ISAL_SHA256_LANE_DATA Struct Reference

Scheduler layer - Lane data.

`#include <sha256_mb.h>`

Collaboration diagram for ISAL_SHA256_LANE_DATA:

9.23.1 Detailed Description

Scheduler layer - Lane data.

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.24 ISAL_SHA256_MB_ARGS_X16 Struct Reference

Scheduler layer - Holds arguments for submitted SHA256 job.

`#include <sha256_mb.h>`

9.24.1 Detailed Description

Scheduler layer - Holds arguments for submitted SHA256 job.

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.25 ISAL_SHA256_MB_JOB_MGR Struct Reference

Scheduler layer - Holds state for multi-buffer SHA256 jobs.

`#include <sha256_mb.h>`

Collaboration diagram for ISAL_SHA256_MB_JOB_MGR:

Data Fields

- uint64_t unused_lanes

9.25.1 Detailed Description

Scheduler layer - Holds state for multi-buffer SHA256 jobs.

9.25.2 Field Documentation

9.25.2.1 unused_lanes

`uint64_t ISAL_SHA256_MB_JOB_MGR::unused_lanes`

each nibble is index (0...3 or 0...7) of unused lanes, nibble 4 or 8 is set to F as a flag

The documentation for this struct was generated from the following file:

- sha256_mb.h

9.26 ISAL_SHA512_HASH_CTX Struct Reference

Context layer - Holds info describing a single SHA512 job for the multi-buffer CTX manager.

`#include <sha512_mb.h>`

Collaboration diagram for ISAL_SHA512_HASH_CTX:

Data Fields

- ISAL_HASH_CTX_STS **status**
  Context status flag.
- ISAL_HASH_CTX_ERROR **error**
  Context error flag.
- uint64_t **total_length**
  Running counter of length processed for this CTX's job.
- const void * **incoming_buffer**
  pointer to data input buffer for this CTX's job
- uint32_t **incoming_buffer_length**
  length of buffer for this job in bytes.
- uint8_t **partial_block_buffer** [ISAL_SHA512_BLOCK_SIZE *2]
  CTX partial blocks.
- void * **user_data**
  pointer for user to keep any job-related data

9.26.1 Detailed Description

Context layer - Holds info describing a single SHA512 job for the multi-buffer CTX manager.

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.27 ISAL_SHA512_HASH_CTX_MGR Struct Reference

Context layer - Holds state for multi-buffer SHA512 jobs.

`#include <sha512_mb.h>`

Collaboration diagram for ISAL_SHA512_HASH_CTX_MGR:

9.27.1 Detailed Description

Context layer - Holds state for multi-buffer SHA512 jobs.

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.28 ISAL_SHA512_JOB Struct Reference

Scheduler layer - Holds info describing a single SHA512 job for the multi-buffer manager.

```
#include <sha512_mb.h>
```

Data Fields

- uint8_t * **buffer**
  pointer to data buffer for this job
- uint64_t **len**
  length of buffer for this job in blocks.
- ISAL_JOB_STS **status**
  output job status
- void * **user_data**
  pointer for user's job-related data

9.28.1 Detailed Description

Scheduler layer - Holds info describing a single SHA512 job for the multi-buffer manager.

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.29 ISAL_SHA512_LANE_DATA Struct Reference

Scheduler layer - Lane data.

```
#include <sha512_mb.h>
```

Collaboration diagram for ISAL_SHA512_LANE_DATA:

9.29.1 Detailed Description

Scheduler layer - Lane data.

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.30 ISAL_SHA512_MB_ARGS_X8 Struct Reference

Scheduler layer - Holds arguments for submitted SHA512 job.

```
#include <sha512_mb.h>
```

9.30.1 Detailed Description

Scheduler layer - Holds arguments for submitted SHA512 job.

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.31 ISAL_SHA512_MB_JOB_MGR Struct Reference

Scheduler layer - Holds state for multi-buffer SHA512 jobs.

```
#include <sha512_mb.h>
```

Collaboration diagram for ISAL_SHA512_MB_JOB_MGR:

Data Fields

- uint64_t unused_lanes

9.31.1 Detailed Description

Scheduler layer - Holds state for multi-buffer SHA512 jobs.

9.31.2 Field Documentation

9.31.2.1 unused_lanes

```
uint64_t ISAL_SHA512_MB_JOB_MGR::unused_lanes
```

each byte is index (00, 01 or 00...03) of unused lanes, byte 2 or 4 is set to FF as a flag

The documentation for this struct was generated from the following file:

- sha512_mb.h

9.32 ISAL_SM3_HASH_CTX Struct Reference

Context layer - Holds info describing a single SM3 job for the multi-buffer CTX manager.

```
#include <sm3_mb.h>
```

Collaboration diagram for ISAL_SM3_HASH_CTX:

Data Fields

- ISAL_HASH_CTX_STS **status**
  Context status flag.
- ISAL_HASH_CTX_ERROR **error**
  Context error flag.
- uint64_t **total_length**
  Running counter of length processed for this CTX's job.
- const void * **incoming_buffer**
  pointer to data input buffer for this CTX's job
- uint32_t **incoming_buffer_length**
  length of buffer for this job in bytes.
- uint8_t **partial_block_buffer** [ISAL_SM3_BLOCK_SIZE *2]
  CTX partial blocks.
- void * **user_data**
  pointer for user to keep any job-related data

9.32.1 Detailed Description

Context layer - Holds info describing a single SM3 job for the multi-buffer CTX manager.

The documentation for this struct was generated from the following file:

- sm3_mb.h

9.33 ISAL_SM3_HASH_CTX_MGR Struct Reference

Context layer - Holds state for multi-buffer SM3 jobs.

`#include <sm3_mb.h>`

Collaboration diagram for ISAL_SM3_HASH_CTX_MGR:

9.33.1 Detailed Description

Context layer - Holds state for multi-buffer SM3 jobs.

The documentation for this struct was generated from the following file:

- sm3_mb.h

9.34 ISAL_SM3_JOB Struct Reference

Scheduler layer - Holds info describing a single SM3 job for the multi-buffer manager.

`#include <sm3_mb.h>`

Data Fields

- uint8_t * **buffer**
  pointer to data buffer for this job
- uint64_t **len**
  length of buffer for this job in blocks.
- ISAL_JOB_STS **status**
  output job status
- void * **user_data**
  pointer for user's job-related data

9.34.1 Detailed Description

Scheduler layer - Holds info describing a single SM3 job for the multi-buffer manager.

The documentation for this struct was generated from the following file:

- sm3_mb.h

9.35 ISAL_SM3_LANE_DATA Struct Reference

Scheduler layer - Lane data.

`#include <sm3_mb.h>`

Collaboration diagram for ISAL_SM3_LANE_DATA:

9.35.1 Detailed Description

Scheduler layer - Lane data.

The documentation for this struct was generated from the following file:

- sm3_mb.h

9.36 ISAL_SM3_MB_ARGS_X16 Struct Reference

Scheduler layer - Holds arguments for submitted SM3 job.

`#include <sm3_mb.h>`

9.36.1 Detailed Description

Scheduler layer - Holds arguments for submitted SM3 job.

The documentation for this struct was generated from the following file:

- sm3_mb.h

9.37 ISAL_SM3_MB_JOB_MGR Struct Reference

Scheduler layer - Holds state for multi-buffer SM3 jobs.

`#include <sm3_mb.h>`

Collaboration diagram for ISAL_SM3_MB_JOB_MGR:

Data Fields

- uint64_t unused_lanes

9.37.1 Detailed Description

Scheduler layer - Holds state for multi-buffer SM3 jobs.

9.37.2 Field Documentation

9.37.2.1 unused_lanes

```
uint64_t ISAL_SM3_MB_JOB_MGR::unused_lanes
```

each nibble is index (0...3 or 0...7) of unused lanes, nibble 4 or 8 is set to F as a flag

The documentation for this struct was generated from the following file:

- sm3_mb.h

Chapter 10

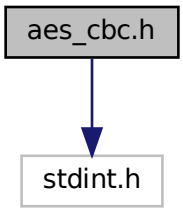
File Documentation

10.1 aes_cbc.h File Reference

AES CBC encryption/decryption function prototypes.

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for aes_cbc.h:



Data Structures

- struct isal_cbc_key_data

  holds intermediate key data used in encryption/decryption

Functions

- int aes_cbc_precomp (uint8_t *key, int key_size, struct isal_cbc_key_data *keys_blk)
  CBC-AES key pre-computation done once for a key.
- void aes_cbc_dec_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 128 bit key Decryption.
- void aes_cbc_dec_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 192 bit key Decryption.
- void aes_cbc_dec_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 256 bit key Decryption.
- int aes_cbc_enc_128 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 128 bit key Encryption.
- int aes_cbc_enc_192 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 192 bit key Encryption.
- int aes_cbc_enc_256 (void *in, uint8_t *IV, uint8_t *keys, void *out, uint64_t len_bytes)
  CBC-AES 256 bit key Encryption.
- int isal_aes_cbc_dec_128 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 128 bit key Decryption.
- int isal_aes_cbc_dec_192 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 192 bit key Decryption.
- int isal_aes_cbc_dec_256 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 256 bit key Decryption.
- int isal_aes_cbc_enc_128 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 128 bit key Encryption.
- int isal_aes_cbc_enc_192 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 192 bit key Encryption.
- int isal_aes_cbc_enc_256 (const void *in, const void *iv, const void *keys, void *out, const uint64_t len_bytes)
  CBC-AES 256 bit key Encryption.

10.1.1 Detailed Description

AES CBC encryption/decryption function prototypes.

10.1.2 Function Documentation

10.1.2.1 aes_cbc_dec_128()

```
void aes_cbc_dec_128 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 128 bit key Decryption.

Deprecated Please use isal_aes_cbc_dec_128() instead.

Requires SSE4.1 and AESNI

arg 1: in: pointer to input (cipher text) arg 2: IV: pointer to IV, Must be 16 bytes aligned to a 16 byte boundary arg 3: keys: pointer to keys, Must be on a 16 byte boundary and length of key size * key rounds arg 4: OUT: pointer to output (plain text ... in-place allowed) arg 5: len_bytes: length in bytes (multiple of 16)

Parameters

in	Input cipher text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size * key rounds or dec_keys of isal_cbc_key_data
out	Output plain text
len_bytes	Must be a multiple of 16 bytes

10.1.2.2 aes_cbc_dec_192()

```
void aes_cbc_dec_192 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 192 bit key Decryption.

Deprecated Please use isal_aes_cbc_dec_192() instead.

Requires SSE4.1 and AESNI

Parameters

in	Input cipher text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size * key rounds or dec_keys of isal_cbc_key_data
out	Output plain text
len_bytes	Must be a multiple of 16 bytes

10.1.2.3 aes_cbc_dec_256()

```
void aes_cbc_dec_256 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 256 bit key Decryption.

Deprecated Please use isal_aes_cbc_dec_256() instead.

Requires SSE4.1 and AESNI

Parameters

in	Input cipher text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size * key rounds or dec_keys of isal_cbc_key_data
out	Output plain text
len_bytes	Must be a multiple of 16 bytes

10.1.2.4 aes_cbc_enc_128()

```
int aes_cbc_enc_128 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 128 bit key Encryption.

Deprecated Please use isal_aes_cbc_enc_128() instead.

Requires SSE4.1 and AESNI

arg 1: in: pointer to input (plain text) arg 2: IV: pointer to IV, Must be 16 bytes aligned to a 16 byte boundary arg 3: keys: pointer to keys, Must be on a 16 byte boundary and length of key size * key rounds arg 4: OUT: pointer to output (cipher text ... in-place allowed) arg 5: len_bytes: length in bytes (multiple of 16)

Parameters

in	Input plain text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size * key rounds or enc_keys of isal_cbc_key_data
out	Output cipher text
len_bytes	Must be a multiple of 16 bytes

10.1.2.5 aes_cbc_enc_192()

```
int aes_cbc_enc_192 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 192 bit key Encryption.

Deprecated Please use isal_aes_cbc_enc_192() instead.

Requires SSE4.1 and AESNI

Parameters

in	Input plain text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size ∗ key rounds or enc_keys of isal_cbc_key_data
out	Output cipher text
len_bytes	Must be a multiple of 16 bytes

10.1.2.6 aes_cbc_enc_256()

```
int aes_cbc_enc_256 (
            void * in,
            uint8_t * IV,
            uint8_t * keys,
            void * out,
            uint64_t len_bytes )
```

CBC-AES 256 bit key Encryption.

Deprecated Please use isal_aes_cbc_enc_256() instead.

Requires SSE4.1 and AESNI

Parameters

in	Input plain text
IV	Must be 16 bytes aligned to a 16 byte boundary
keys	Must be on a 16 byte boundary and length of key size ∗ key rounds or enc_keys of isal_cbc_key_data
out	Output cipher text
len_bytes	Must be a multiple of 16 bytes

10.1.2.7 aes_cbc_precomp()

```
int aes_cbc_precomp (
            uint8_t * key,
            int key_size,
            struct isal_cbc_key_data * keys_blk )
```

CBC-AES key pre-computation done once for a key.

Deprecated Please use isal_aes_keyexp_128(), isal_aes_keyexp_192() or isal_aes_keyexp_256() instead.

Requires SSE4.1 and AESNI

arg 1: in: pointer to key arg 2: OUT: pointer to a key expanded data

10.1.2.8 isal_aes_cbc_dec_128()

```
int isal_aes_cbc_dec_128 (
            const void * in,
            const void * iv,
            const void * keys,
            void * out,
            const uint64_t len_bytes )
```

CBC-AES 128 bit key Decryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input ciphertext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output plaintext
len_bytes	Input length. Must be a multiple of 16 bytes

10.1.2.9 isal_aes_cbc_dec_192()

```
int isal_aes_cbc_dec_192 (
            const void * in,
            const void * iv,
            const void * keys,
```

```
            void * out,
            const uint64_t len_bytes )
```

CBC-AES 192 bit key Decryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input ciphertext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output plaintext
len_bytes	Input length. Must be a multiple of 16 bytes

10.1.2.10 isal_aes_cbc_dec_256()

```
int isal_aes_cbc_dec_256 (
            const void * in,
            const void * iv,
            const void * keys,
            void * out,
            const uint64_t len_bytes )
```

CBC-AES 256 bit key Decryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input ciphertext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output plaintext
len_bytes	Input length. Must be a multiple of 16 bytes

10.1.2.11 isal_aes_cbc_enc_128()

```
int isal_aes_cbc_enc_128 (
             const void * in,
             const void * iv,
             const void * keys,
             void * out,
             const uint64_t len_bytes )
```

CBC-AES 128 bit key Encryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input plaintext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output ciphertext
len_bytes	Input length. Must be a multiple of 16 bytes

10.1.2.12 isal_aes_cbc_enc_192()

```
int isal_aes_cbc_enc_192 (
             const void * in,
```

```
            const void * iv,
            const void * keys,
            void * out,
            const uint64_t len_bytes )
```

CBC-AES 192 bit key Encryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input plaintext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output ciphertext
len_bytes	Input length. Must be a multiple of 16 bytes

10.1.2.13 isal_aes_cbc_enc_256()

```
int isal_aes_cbc_enc_256 (
            const void * in,
            const void * iv,
            const void * keys,
            void * out,
            const uint64_t len_bytes )
```

CBC-AES 256 bit key Encryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

in	Input plaintext
iv	Initialization vector. Must be 16 bytes aligned.
keys	Expanded decryption keys. Must be on a 16 byte boundary.
out	Output ciphertext
len_bytes	Input length. Must be a multiple of 16 bytes

10.2 aes_cbc.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00035 #ifndef _AES_CBC_h
00036 #define _AES_CBC_h
00037
00038 #include <stdint.h>
00039
00040 #include "types.h"
00041
00042 #ifdef __cplusplus
00043 extern "C" {
00044
00045 #endif
00046
00047 /*
00048  * Define enums from API v2.24, so applications that were using this version
00049  * will still be compiled successfully.
00050  * This list does not need to be extended for new definitions.
00051  */
00052 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00053 /***** Previous hash constants and typedefs *****/
```

```
00054 #define CBC_128_BITS ISAL_CBC_128_BITS
00055 #define CBC_192_BITS ISAL_CBC_192_BITS
00056 #define CBC_256_BITS ISAL_CBC_256_BITS
00057
00058 #define CBC_ROUND_KEY_LEN  ISAL_CBC_ROUND_KEY_LEN
00059 #define CBC_128_KEY_ROUNDS ISAL_CBC_128_KEY_ROUNDS
00060 #define CBC_192_KEY_ROUNDS ISAL_CBC_192_KEY_ROUNDS
00061 #define CBC_256_KEY_ROUNDS ISAL_CBC_256_KEY_ROUNDS
00062 #define CBC_MAX_KEYS_SIZE  ISAL_CBC_MAX_KEYS_SIZE
00063
00064 #define CBC_IV_DATA_LEN ISAL_CBC_IV_DATA_LEN
00065
00066 #define cbc_key_data isal_cbc_key_data
00067 #define cbc_key_size isal_cbc_key_size
00068 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00069
00070 typedef enum isal_cbc_key_size {
00071         ISAL_CBC_128_BITS = 16,
00072         ISAL_CBC_192_BITS = 24,
00073         ISAL_CBC_256_BITS = 32
00074 } isal_cbc_key_size;
00075 #define ISAL_CBC_ROUND_KEY_LEN  (16)
00076 #define ISAL_CBC_128_KEY_ROUNDS (10 + 1) /*expanded key holds 10 key rounds plus original key*/
00077 #define ISAL_CBC_192_KEY_ROUNDS (12 + 1) /*expanded key holds 12 key rounds plus original key*/
00078 #define ISAL_CBC_256_KEY_ROUNDS (14 + 1) /*expanded key holds 14 key rounds plus original key*/
00079 #define ISAL_CBC_MAX_KEYS_SIZE  (ISAL_CBC_ROUND_KEY_LEN * ISAL_CBC_256_KEY_ROUNDS)
00080
00081 #define ISAL_CBC_IV_DATA_LEN (16)
00082
00086 struct isal_cbc_key_data { // must be 16 byte aligned
00087         uint8_t enc_keys[ISAL_CBC_MAX_KEYS_SIZE];
00088         uint8_t dec_keys[ISAL_CBC_MAX_KEYS_SIZE];
00089 };
00090
00100 ISAL_DEPRECATED("Please use isal_aes_keyexp_128/192/256() instead")
00101 int
00102 aes_cbc_precomp(uint8_t *key, int key_size, struct isal_cbc_key_data *keys_blk);
00103
00115 ISAL_DEPRECATED("Please use isal_aes_cbc_dec_128() instead")
00116 void
00117 aes_cbc_dec_128(void *in,
00118                 uint8_t *IV,
00119                 uint8_t *keys,
00121                 void *out,
00122                 uint64_t len_bytes
00123 );
00124
00131 ISAL_DEPRECATED("Please use isal_aes_cbc_dec_192() instead")
00132 void
00133 aes_cbc_dec_192(void *in,
00134                 uint8_t *IV,
00135                 uint8_t *keys,
00137                 void *out,
00138                 uint64_t len_bytes
00139 );
00140
00147 ISAL_DEPRECATED("Please use isal_aes_cbc_dec_256() instead")
00148 void
00149 aes_cbc_dec_256(void *in,
00150                 uint8_t *IV,
00151                 uint8_t *keys,
00153                 void *out,
00154                 uint64_t len_bytes
00155 );
00156
00168 ISAL_DEPRECATED("Please use isal_aes_cbc_enc_128() instead")
00169 int
00170 aes_cbc_enc_128(void *in,
00171                 uint8_t *IV,
00172                 uint8_t *keys,
00174                 void *out,
00175                 uint64_t len_bytes
00176 );
00183 ISAL_DEPRECATED("Please use isal_aes_cbc_enc_192() instead")
00184 int
00185 aes_cbc_enc_192(void *in,
00186                 uint8_t *IV,
00187                 uint8_t *keys,
00189                 void *out,
00190                 uint64_t len_bytes
00191 );
```

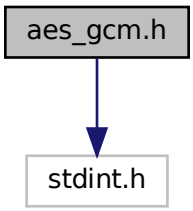
```
00192
00199 ISAL_DEPRECATED("Please use isal_aes_cbc_enc_256() instead")
00200 int
00201 aes_cbc_enc_256(void *in,
00202                  uint8_t *IV,
00203                  uint8_t *keys,
00205                  void *out,
00206                  uint64_t len_bytes
00207 );
00208
00217 int
00218 isal_aes_cbc_dec_128(const void *in,
00219                      const void *iv,
00220                      const void *keys,
00221                      void *out,
00222                      const uint64_t len_bytes
00223 );
00224
00232 int
00233 isal_aes_cbc_dec_192(const void *in,
00234                      const void *iv,
00235                      const void *keys,
00236                      void *out,
00237                      const uint64_t len_bytes
00238 );
00239
00247 int
00248 isal_aes_cbc_dec_256(const void *in,
00249                      const void *iv,
00250                      const void *keys,
00251                      void *out,
00252                      const uint64_t len_bytes
00253 );
00254
00263 int
00264 isal_aes_cbc_enc_128(const void *in,
00265                      const void *iv,
00266                      const void *keys,
00267                      void *out,
00268                      const uint64_t len_bytes
00269 );
00277 int
00278 isal_aes_cbc_enc_192(const void *in,
00279                      const void *iv,
00280                      const void *keys,
00281                      void *out,
00282                      const uint64_t len_bytes
00283 );
00284
00292 int
00293 isal_aes_cbc_enc_256(const void *in,
00294                      const void *iv,
00295                      const void *keys,
00296                      void *out,
00297                      const uint64_t len_bytes
00298 );
00299
00300 #ifdef __cplusplus
00301 }
00302 #endif //__cplusplus
00303 #endif // ifndef _AES_CBC_h
```

10.3 aes_gcm.h File Reference

AES GCM encryption/decryption function prototypes.

```
#include "types.h"
#include <stdint.h>
```

Include dependency graph for aes_gcm.h:



Data Structures

- struct isal_gcm_key_data

  holds intermediate key data needed to improve performance
- struct isal_gcm_context_data

  holds GCM operation context

Functions

- void aes_gcm_enc_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Encryption using 128 bit keys.
- void aes_gcm_enc_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Encryption using 256 bit keys.
- void aes_gcm_dec_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys.
- void aes_gcm_dec_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys.
- void aes_gcm_init_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

  Start a AES-GCM Encryption message 128 bit key.
- void aes_gcm_init_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *iv, uint8_t const *aad, uint64_t aad_len)

  Start a AES-GCM Encryption message 256 bit key.

- void aes_gcm_enc_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Encrypt a block of a AES-128-GCM Encryption message.
- void aes_gcm_enc_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Encrypt a block of a AES-256-GCM Encryption message.
- void aes_gcm_dec_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Decrypt a block of a AES-128-GCM Encryption message.
- void aes_gcm_dec_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Decrypt a block of a AES-256-GCM Encryption message.
- void aes_gcm_enc_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

  End encryption of a AES-128-GCM Encryption message.
- void aes_gcm_enc_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

  End encryption of a AES-256-GCM Encryption message.
- void aes_gcm_dec_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

  End decryption of a AES-128-GCM Encryption message.
- void aes_gcm_dec_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, uint64_t auth_tag_len)

  End decryption of a AES-256-GCM Encryption message.
- void aes_gcm_pre_128 (const void *key, struct isal_gcm_key_data *key_data)

  Pre-processes GCM key data 128 bit.
- void aes_gcm_pre_256 (const void *key, struct isal_gcm_key_data *key_data)

  Pre-processes GCM key data 128 bit.
- void aes_gcm_enc_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Encryption using 128 bit keys, Non-temporal data.
- void aes_gcm_enc_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Encryption using 256 bit keys, Non-temporal data.
- void aes_gcm_dec_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys, Non-temporal data.
- void aes_gcm_dec_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context↩_data, uint8_t *out, uint8_t const *in, uint64_t len, uint8_t *iv, uint8_t const *aad, uint64_t aad_len, uint8_t *auth_tag, uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys, Non-temporal data.
- void aes_gcm_enc_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Encrypt a block of a AES-128-GCM Encryption message, Non-temporal data.
- void aes_gcm_enc_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

Encrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

- void aes_gcm_dec_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Decrypt a block of a AES-128-GCM Encryption message, Non-temporal data.
- void aes_gcm_dec_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, uint64_t len)

  Decrypt a block of a AES-256-GCM Encryption message, Non-temporal data.
- int isal_aes_gcm_enc_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Encryption using 128 bit keys.
- int isal_aes_gcm_enc_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Encryption using 256 bit keys.
- int isal_aes_gcm_dec_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys.
- int isal_aes_gcm_dec_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys.
- int isal_aes_gcm_init_128 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len)

  Start a AES-GCM Encryption message 128 bit key.
- int isal_aes_gcm_init_256 (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len)

  Start a AES-GCM Encryption message 256 bit key.
- int isal_aes_gcm_enc_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Encrypt a block of a AES-128-GCM Encryption message.
- int isal_aes_gcm_enc_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Encrypt a block of a AES-256-GCM Encryption message.
- int isal_aes_gcm_dec_128_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Decrypt a block of a AES-128-GCM Encryption message.
- int isal_aes_gcm_dec_256_update (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Decrypt a block of a AES-256-GCM Encryption message.
- int isal_aes_gcm_enc_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

  End encryption of a AES-128-GCM Encryption message.
- int isal_aes_gcm_enc_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

  End encryption of a AES-256-GCM Encryption message.
- int isal_aes_gcm_dec_128_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

End decryption of a AES-128-GCM Encryption message.

- int isal_aes_gcm_dec_256_finalize (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *auth_tag, const uint64_t auth_tag_len)

  End decryption of a AES-256-GCM Encryption message.

- int isal_aes_gcm_pre_128 (const void *key, struct isal_gcm_key_data *key_data)

  Pre-processes GCM key data 128 bit.

- int isal_aes_gcm_pre_256 (const void *key, struct isal_gcm_key_data *key_data)

  Pre-processes GCM key data 256 bit.

- int isal_aes_gcm_enc_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Encryption using 128 bit keys, Non-temporal data.

- int isal_aes_gcm_enc_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Encryption using 256 bit keys, Non-temporal data.

- int isal_aes_gcm_dec_128_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys, Non-temporal data.

- int isal_aes_gcm_dec_256_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len, const uint8_t *iv, const uint8_t *aad, const uint64_t aad_len, uint8_t *auth_tag, const uint64_t auth_tag_len)

  GCM-AES Decryption using 128 bit keys, Non-temporal data.

- int isal_aes_gcm_enc_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Encrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

- int isal_aes_gcm_enc_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Encrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

- int isal_aes_gcm_dec_128_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Decrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

- int isal_aes_gcm_dec_256_update_nt (const struct isal_gcm_key_data *key_data, struct isal_gcm_context_data *context_data, uint8_t *out, const uint8_t *in, const uint64_t len)

  Decrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

10.3.1 Detailed Description

AES GCM encryption/decryption function prototypes.

At build time there is an option to use non-temporal loads and stores selected by defining the compile time option NT_LDST. The use of this option places the following restriction on the gcm encryption functions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- When using the streaming API, all partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended.

10.3.2 Function Documentation

10.3.2.1 aes_gcm_dec_128()

```
void aes_gcm_dec_128 (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             uint8_t const * in,
             uint64_t len,
             uint8_t * iv,
             uint8_t const * aad,
             uint64_t aad_len,
             uint8_t * auth_tag,
             uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys.

Deprecated Please use isal_aes_gcm_dec_128() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.2 aes_gcm_dec_128_finalize()

```
void aes_gcm_dec_128_finalize (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * auth_tag,
             uint64_t auth_tag_len )
```

End decryption of a AES-128-GCM Encryption message.

Deprecated Please use isal_aes_gcm_dec_128_finalize() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.3 aes_gcm_dec_128_nt()

```
void aes_gcm_dec_128_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            uint8_t const * in,
            uint64_t len,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys, Non-temporal data.

Non-temporal version of decrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

  Deprecated Please use isal_aes_gcm_dec_128_nt() instead.

  Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.4 aes_gcm_dec_128_update()

```
void aes_gcm_dec_128_update (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Decrypt a block of a AES-128-GCM Encryption message.

Deprecated Please use isal_aes_gcm_dec_128_update() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.5 aes_gcm_dec_128_update_nt()

```
void aes_gcm_dec_128_update_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Decrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

Non-temporal version of decrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

  Deprecated Please use isal_aes_gcm_dec_128_update_nt() instead.

  Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.6 aes_gcm_dec_256()

```
void aes_gcm_dec_256 (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             uint8_t const * in,
             uint64_t len,
             uint8_t * iv,
             uint8_t const * aad,
             uint64_t aad_len,
             uint8_t * auth_tag,
             uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys.

Deprecated Please use isal_aes_gcm_dec_256() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.7 aes_gcm_dec_256_finalize()

```
void aes_gcm_dec_256_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

End decryption of a AES-256-GCM Encryption message.

Deprecated Please use isal_aes_gcm_dec_256_finalize() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.8 aes_gcm_dec_256_nt()

```
void aes_gcm_dec_256_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            uint8_t const * in,
            uint64_t len,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys, Non-temporal data.

Non-temporal version of decrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

  Deprecated Please use isal_aes_gcm_dec_256_nt() instead.

  Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.9 aes_gcm_dec_256_update()

```
void aes_gcm_dec_256_update (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Decrypt a block of a AES-256-GCM Encryption message.

Deprecated Please use isal_aes_gcm_dec_256_update() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.10 aes_gcm_dec_256_update_nt()

```
void aes_gcm_dec_256_update_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
```

```
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Decrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

Non-temporal version of decrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Deprecated Please use isal_aes_gcm_dec_256_update_nt() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.11 aes_gcm_enc_128()

```
void aes_gcm_enc_128 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            uint8_t const * in,
            uint64_t len,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

GCM-AES Encryption using 128 bit keys.

Deprecated Please use isal_aes_gcm_enc_128() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.12 aes_gcm_enc_128_finalize()

```
void aes_gcm_enc_128_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

End encryption of a AES-128-GCM Encryption message.

Deprecated Please use isal_aes_gcm_enc_128_finalize() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.13 aes_gcm_enc_128_nt()

```
void aes_gcm_enc_128_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            uint8_t const * in,
```

```
            uint64_t len,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

GCM-AES Encryption using 128 bit keys, Non-temporal data.

Non-temporal version of encrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Deprecated Please use isal_aes_gcm_enc_128_nt() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.14 aes_gcm_enc_128_update()

```
void aes_gcm_enc_128_update (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Encrypt a block of a AES-128-GCM Encryption message.

Deprecated Please use isal_aes_gcm_enc_128_update() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.15 aes_gcm_enc_128_update_nt()

```
void aes_gcm_enc_128_update_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            uint64_t len )
```

Encrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

Non-temporal version of encrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Deprecated Please use isal_aes_gcm_enc_128_update_nt() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.16 aes_gcm_enc_256()

```
void aes_gcm_enc_256 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
```

```
            uint8_t const * in,
            uint64_t len,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

GCM-AES Encryption using 256 bit keys.

Deprecated Please use isal_aes_gcm_enc_256() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.17 aes_gcm_enc_256_finalize()

```
void aes_gcm_enc_256_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            uint64_t auth_tag_len )
```

End encryption of a AES-256-GCM Encryption message.

Deprecated Please use isal_aes_gcm_enc_256_finalize() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.18 aes_gcm_enc_256_nt()

```
void aes_gcm_enc_256_nt (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             uint8_t const * in,
             uint64_t len,
             uint8_t * iv,
             uint8_t const * aad,
             uint64_t aad_len,
             uint8_t * auth_tag,
             uint64_t auth_tag_len )
```

GCM-AES Encryption using 256 bit keys, Non-temporal data.

Non-temporal version of encrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Deprecated Please use isal_aes_gcm_enc_256_nt() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.19 aes_gcm_enc_256_update()

```
void aes_gcm_enc_256_update (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             uint64_t len )
```

Encrypt a block of a AES-256-GCM Encryption message.

Deprecated Please use isal_aes_gcm_enc_256_update() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.20 aes_gcm_enc_256_update_nt()

```
void aes_gcm_enc_256_update_nt (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             uint64_t len )
```

Encrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

Non-temporal version of encrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

  Deprecated Please use isal_aes_gcm_enc_256_update_nt() instead.

  Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.21 aes_gcm_init_128()

```
void aes_gcm_init_128 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len )
```

Start a AES-GCM Encryption message 128 bit key.

Deprecated Please use isal_aes_gcm_init_128() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
iv	Pointer to 12 byte IV structure Internally, library concates 0x00000001 value to it
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD

10.3.2.22 aes_gcm_init_256()

```
void aes_gcm_init_256 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * iv,
            uint8_t const * aad,
            uint64_t aad_len )
```

Start a AES-GCM Encryption message 256 bit key.

Deprecated Please use isal_aes_gcm_init_256() instead.

Requires SSE4.1 and AESNI

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
iv	Pointer to 12 byte IV structure Internally, library concates 0x00000001 value to it
aad	Additional Authentication Data (AAD)
aad_len	Length of AAD

10.3.2.23 aes_gcm_pre_128()

```
void aes_gcm_pre_128 (
             const void * key,
             struct isal_gcm_key_data * key_data )
```

Pre-processes GCM key data 128 bit.

Prefills the gcm key data with key values for each round and the initial sub hash key for tag encoding

Deprecated Please use isal_aes_gcm_pre_128() instead.

Requires SSE4.1 and AESNI

Parameters

key	Pointer to key data
key_data	GCM expanded key data

10.3.2.24 aes_gcm_pre_256()

```
void aes_gcm_pre_256 (
             const void * key,
             struct isal_gcm_key_data * key_data )
```

Pre-processes GCM key data 128 bit.

Prefills the gcm key data with key values for each round and the initial sub hash key for tag encoding

Deprecated Please use isal_aes_gcm_pre_256() instead.

Requires SSE4.1 and AESNI

Parameters

key	Pointer to key data
key_data	GCM expanded key data

10.3.2.25 isal_aes_gcm_dec_128()

```
int isal_aes_gcm_dec_128 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.26 isal_aes_gcm_dec_128_finalize()

```
int isal_aes_gcm_dec_128_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

End decryption of a AES-128-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.27 isal_aes_gcm_dec_128_nt()

```
int isal_aes_gcm_dec_128_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys, Non-temporal data.

Non-temporal version of decrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.28 isal_aes_gcm_dec_128_update()

```
int isal_aes_gcm_dec_128_update (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             const uint64_t len )
```

Decrypt a block of a AES-128-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.29 isal_aes_gcm_dec_128_update_nt()

```
int isal_aes_gcm_dec_128_update_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len )
```

Decrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

Non-temporal version of decrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data

Parameters

context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.30 isal_aes_gcm_dec_256()

```
int isal_aes_gcm_dec_256 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output

Parameters

auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.31 isal_aes_gcm_dec_256_finalize()

```
int isal_aes_gcm_dec_256_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

End decryption of a AES-256-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.32 isal_aes_gcm_dec_256_nt()

```
int isal_aes_gcm_dec_256_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
```

```
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Decryption using 128 bit keys, Non-temporal data.

Non-temporal version of decrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed
in	Ciphertext input
len	Length of data in Bytes for decryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.33 isal_aes_gcm_dec_256_update()

```
int isal_aes_gcm_dec_256_update (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len )
```

Decrypt a block of a AES-256-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.34 isal_aes_gcm_dec_256_update_nt()

```
int isal_aes_gcm_dec_256_update_nt (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             const uint64_t len )
```

Decrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

Non-temporal version of decrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Plaintext output. Decrypt in-place is allowed.
in	Ciphertext input
len	Length of data in Bytes for decryption

10.3.2.35 isal_aes_gcm_enc_128()

```
int isal_aes_gcm_enc_128 (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             const uint64_t len,
             const uint8_t * iv,
             const uint8_t * aad,
             const uint64_t aad_len,
             uint8_t * auth_tag,
             const uint64_t auth_tag_len )
```

GCM-AES Encryption using 128 bit keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed

Parameters

in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.36 isal_aes_gcm_enc_128_finalize()

```
int isal_aes_gcm_enc_128_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

End encryption of a AES-128-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.37 isal_aes_gcm_enc_128_nt()

```
int isal_aes_gcm_enc_128_nt (
```

```
const struct isal_gcm_key_data * key_data,
struct isal_gcm_context_data * context_data,
uint8_t * out,
const uint8_t * in,
const uint64_t len,
const uint8_t * iv,
const uint8_t * aad,
const uint64_t aad_len,
uint8_t * auth_tag,
const uint64_t auth_tag_len )
```

GCM-AES Encryption using 128 bit keys, Non-temporal data.

Non-temporal version of encrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.38 isal_aes_gcm_enc_128_update()

```
int isal_aes_gcm_enc_128_update (
```

```
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len )
```

Encrypt a block of a AES-128-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.39 isal_aes_gcm_enc_128_update_nt()

```
int isal_aes_gcm_enc_128_update_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len )
```

Encrypt a block of a AES-128-GCM Encryption message, Non-temporal data.

Non-temporal version of encrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.40 isal_aes_gcm_enc_256()

```
int isal_aes_gcm_enc_256 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Encryption using 256 bit keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.41 isal_aes_gcm_enc_256_finalize()

```
int isal_aes_gcm_enc_256_finalize (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

End encryption of a AES-256-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.42 isal_aes_gcm_enc_256_nt()

```
int isal_aes_gcm_enc_256_nt (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            uint8_t * out,
            const uint8_t * in,
            const uint64_t len,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len,
            uint8_t * auth_tag,
            const uint64_t auth_tag_len )
```

GCM-AES Encryption using 256 bit keys, Non-temporal data.

Non-temporal version of encrypt has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed
in	Plaintext input
len	Length of data in Bytes for encryption
iv	iv pointer to 12 byte IV structure. Internally, library concates 0x00000001 value to it.
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD
auth_tag	Authenticated Tag output
auth_tag_len	Authenticated Tag Length in bytes (must be a multiple of 4 bytes). Valid values are 16 (most likely), 12 or 8

10.3.2.43 isal_aes_gcm_enc_256_update()

```
int isal_aes_gcm_enc_256_update (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             const uint64_t len )
```

Encrypt a block of a AES-256-GCM Encryption message.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.44 isal_aes_gcm_enc_256_update_nt()

```
int isal_aes_gcm_enc_256_update_nt (
             const struct isal_gcm_key_data * key_data,
             struct isal_gcm_context_data * context_data,
             uint8_t * out,
             const uint8_t * in,
             const uint64_t len )
```

Encrypt a block of a AES-256-GCM Encryption message, Non-temporal data.

Non-temporal version of encrypt update has additional restrictions:

- The plaintext and ciphertext buffers must be aligned on a 64 byte boundary.

- All partial input buffers must be a multiple of 64 bytes long except for the last input buffer.
- In-place encryption/decryption is not recommended. Performance can be slow.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
out	Ciphertext output. Encrypt in-place is allowed.
in	Plaintext input
len	Length of data in Bytes for encryption

10.3.2.45 isal_aes_gcm_init_128()

```
int isal_aes_gcm_init_128 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len )
```

Start a AES-GCM Encryption message 128 bit key.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
iv	Pointer to 12 byte IV structure Internally, library concates 0x00000001 value to it
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD

10.3.2.46 isal_aes_gcm_init_256()

```
int isal_aes_gcm_init_256 (
            const struct isal_gcm_key_data * key_data,
            struct isal_gcm_context_data * context_data,
            const uint8_t * iv,
            const uint8_t * aad,
            const uint64_t aad_len )
```

Start a AES-GCM Encryption message 256 bit key.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key_data	GCM expanded key data
context_data	GCM operation context data
iv	Pointer to 12 byte IV structure Internally, library concates 0x00000001 value to it
aad	Additional Authenticated Data (AAD)
aad_len	Length of AAD

10.3.2.47 isal_aes_gcm_pre_128()

```
int isal_aes_gcm_pre_128 (
            const void * key,
            struct isal_gcm_key_data * key_data )
```

Pre-processes GCM key data 128 bit.

Prefills the gcm key data with key values for each round and the initial sub hash key for tag encoding

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key	Pointer to key data
key_data	GCM expanded key data

10.3.2.48 isal_aes_gcm_pre_256()

```
int isal_aes_gcm_pre_256 (
            const void * key,
            struct isal_gcm_key_data * key_data )
```

Pre-processes GCM key data 256 bit.

Prefills the gcm key data with key values for each round and the initial sub hash key for tag encoding

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key	Pointer to key data
key_data	GCM expanded key data

10.4 aes_gcm.h

Go to the documentation of this file.

```
00001 /*****************************************************************************
00002   Copyright(c) 2011-2024 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 *****************************************************************************/
00029
00030 #include "types.h"
00031
00049 /*
00050 ; References:
00051 ;       This code was derived and highly optimized from the code described in paper:
00052 ;               Vinodh Gopal et. al. Optimized Galois-Counter-Mode Implementation on Intel
00053 Architecture Processors. August, 2010
00054 ;
00055 ;       For the shift-based reductions used in this code, we used the method described in paper:
00056 ;               Shay Gueron, Michael E. Kounavis. Intel Carry-Less Multiplication Instruction and
00057 its Usage for Computing the GCM Mode. January, 2010.
00058 ;
00059 ;
00060 ;
00061 ; Assumptions: Support for SSE4.1 or greater, AVX or AVX2
00062 ;
00063 ;
00064 ; iv:
00065 ;       0                   1                   2                   3
00066 ;       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
00067 ;       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
00068 ;       |                             Salt  (From the SA)               |
00069 ;       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
00070 ;       |                     Initialization Vector                     |
00071 ;       |         (This is the sequence number from IPSec header)       |
00072 ;       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
00073 ;       |                              0x1                              |
00074 ;       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
00075 ;
00076 ; TLen:
00077 ;       from the definition of the spec, TLen can only be 8, 12 or 16 bytes.
00078 ;
00079  */
00080 #ifndef _AES_GCM_h
```

```
00081 #define _AES_GCM_h
00082
00083 #include <stdint.h>
00084
00085 #ifdef __cplusplus
00086 extern "C" {
00087 #endif
00088
00089 /*
00090  * Define enums from API v2.24, so applications that were using this version
00091  * will still be compiled successfully.
00092  * This list does not need to be extended for new definitions.
00093  */
00094 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00095 /***** Previous hash constants and typedefs *****/
00096 #define MAX_TAG_LEN ISAL_GCM_MAX_TAG_LEN
00097
00098 #define GCM_IV_LEN       (16)
00099 #define GCM_IV_DATA_LEN  ISAL_GCM_IV_LEN
00100 #define GCM_IV_END_MARK  { 0x00, 0x00, 0x00, 0x01 };
00101 #define GCM_IV_END_START (12)
00102
00103 #define GCM_128_KEY_LEN ISAL_GCM_128_KEY_LEN
00104 #define GCM_256_KEY_LEN ISAL_GCM_256_KEY_LEN
00105
00106 #define GCM_BLOCK_LEN   ISAL_GCM_BLOCK_LEN
00107 #define GCM_ENC_KEY_LEN ISAL_GCM_ENC_KEY_LEN
00108 #define GCM_KEY_SETS    ISAL_GCM_KEY_SETS
00109
00110 #define GCM_MAX_LEN ISAL_GCM_MAX_LEN
00111
00112 #define LONGEST_TESTED_AAD_LENGTH (2 * 1024)
00113
00114 #define gcm_key_data     isal_gcm_key_data
00115 #define gcm_context_data isal_gcm_context_data
00116 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00117
00118 /* Authenticated Tag Length in bytes. Valid values are 16 (most likely), 12 or 8. */
00119 #define ISAL_GCM_MAX_TAG_LEN (16)
00120 //
00121 // IV data is limited to 12 bytes.
00122 //
00123 #define ISAL_GCM_IV_LEN (12)
00124
00125 // Key lengths of 128 and 256 supported
00126 #define ISAL_GCM_128_KEY_LEN (16)
00127 #define ISAL_GCM_256_KEY_LEN (32)
00128
00129 #define ISAL_GCM_BLOCK_LEN   16
00130 #define ISAL_GCM_ENC_KEY_LEN 16
00131 #define ISAL_GCM_KEY_SETS    (15) /*exp key + 14 exp round keys */
00132
00133 #define ISAL_GCM_MAX_LEN UINT64_C(((1ULL « 39) - 256) - 1)
00134
00140 #ifdef __WIN32
00141 __declspec(align(16))
00142 #endif /* WIN32 */
00143 struct isal_gcm_key_data {
00144         uint8_t expanded_keys[ISAL_GCM_ENC_KEY_LEN * ISAL_GCM_KEY_SETS];
00145         uint8_t shifted_hkey_1[ISAL_GCM_ENC_KEY_LEN];   // store HashKey «1 mod poly here
00146         uint8_t shifted_hkey_2[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^2 «1 mod poly here
00147         uint8_t shifted_hkey_3[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^3 «1 mod poly here
00148         uint8_t shifted_hkey_4[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^4 «1 mod poly here
00149         uint8_t shifted_hkey_5[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^5 «1 mod poly here
00150         uint8_t shifted_hkey_6[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^6 «1 mod poly here
00151         uint8_t shifted_hkey_7[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^7 «1 mod poly here
00152         uint8_t shifted_hkey_8[ISAL_GCM_ENC_KEY_LEN];   // store HashKey^8 «1 mod poly here
00153         uint8_t shifted_hkey_1_k[ISAL_GCM_ENC_KEY_LEN]; // store XOR of High 64 bits
00154         uint8_t shifted_hkey_2_k[ISAL_GCM_ENC_KEY_LEN]; // and Low 64b of HashKey^n «1 mod poly
00155         uint8_t shifted_hkey_3_k[ISAL_GCM_ENC_KEY_LEN]; // here (for Karatsuba purposes)
00156         uint8_t shifted_hkey_4_k[ISAL_GCM_ENC_KEY_LEN];
00157         uint8_t shifted_hkey_5_k[ISAL_GCM_ENC_KEY_LEN];
00158         uint8_t shifted_hkey_6_k[ISAL_GCM_ENC_KEY_LEN];
00159         uint8_t shifted_hkey_7_k[ISAL_GCM_ENC_KEY_LEN];
00160         uint8_t shifted_hkey_8_k[ISAL_GCM_ENC_KEY_LEN];
00161         uint8_t shifted_hkey_n_k[ISAL_GCM_ENC_KEY_LEN *
00162                                  (64 - 16)]; // Others vaes version needs 2x32
00163 }
00164 #if defined(__unix__) || (__MINGW32__)
00165 __attribute__((aligned(16)));
00166 #else
```

```
00167 ;
00168 #endif
00169
00173 struct isal_gcm_context_data {
00174         // init, update and finalize context data
00175         uint8_t aad_hash[ISAL_GCM_BLOCK_LEN];
00176         uint64_t aad_length;
00177         uint64_t in_length;
00178         uint8_t partial_block_enc_key[ISAL_GCM_BLOCK_LEN];
00179         uint8_t orig_IV[ISAL_GCM_BLOCK_LEN];
00180         uint8_t current_counter[ISAL_GCM_BLOCK_LEN];
00181         uint64_t partial_block_length;
00182 };
00183
00184 /* ------------------ New interface for separate expanded keys ------------ */
00185
00192 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_128() instead.")
00193 void
00194 aes_gcm_enc_128(const struct isal_gcm_key_data *key_data,
00195                 struct isal_gcm_context_data *context_data,
00196                 uint8_t *out,
00197                 uint8_t const *in,
00198                 uint64_t len,
00199                 uint8_t *iv,
00201                 uint8_t const *aad,
00202                 uint64_t aad_len,
00203                 uint8_t *auth_tag,
00204                 uint64_t auth_tag_len
00207 );
00208
00215 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_256() instead.")
00216 void
00217 aes_gcm_enc_256(const struct isal_gcm_key_data *key_data,
00218                 struct isal_gcm_context_data *context_data,
00219                 uint8_t *out,
00220                 uint8_t const *in,
00221                 uint64_t len,
00222                 uint8_t *iv,
00224                 uint8_t const *aad,
00225                 uint64_t aad_len,
00226                 uint8_t *auth_tag,
00227                 uint64_t auth_tag_len
00230 );
00231
00238 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_128() instead.")
00239 void
00240 aes_gcm_dec_128(const struct isal_gcm_key_data *key_data,
00241                 struct isal_gcm_context_data *context_data,
00242                 uint8_t *out,
00243                 uint8_t const *in,
00244                 uint64_t len,
00245                 uint8_t *iv,
00247                 uint8_t const *aad,
00248                 uint64_t aad_len,
00249                 uint8_t *auth_tag,
00250                 uint64_t auth_tag_len
00253 );
00254
00261 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_256() instead.")
00262 void
00263 aes_gcm_dec_256(const struct isal_gcm_key_data *key_data,
00264                 struct isal_gcm_context_data *context_data,
00265                 uint8_t *out,
00266                 uint8_t const *in,
00267                 uint64_t len,
00268                 uint8_t *iv,
00270                 uint8_t const *aad,
00271                 uint64_t aad_len,
00272                 uint8_t *auth_tag,
00273                 uint64_t auth_tag_len
00276 );
00277
00284 ISAL_DEPRECATED("Please use isal_aes_gcm_init_128() instead.")
00285 void
00286 aes_gcm_init_128(const struct isal_gcm_key_data *key_data,
00287                  struct isal_gcm_context_data *context_data,
00288                  uint8_t *iv,
00290                  uint8_t const *aad,
00291                  uint64_t aad_len
00292 );
00293
```

```
00300 ISAL_DEPRECATED("Please use isal_aes_gcm_init_256() instead.")
00301 void
00302 aes_gcm_init_256(const struct isal_gcm_key_data *key_data,
00303                  struct isal_gcm_context_data *context_data,
00304                  uint8_t *iv,
00306                  uint8_t const *aad,
00307                  uint64_t aad_len
00308 );
00309
00316 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_128_update() instead.")
00317 void
00318 aes_gcm_enc_128_update(const struct isal_gcm_key_data *key_data,
00319                        struct isal_gcm_context_data *context_data,
00320                        uint8_t *out,
00321                        const uint8_t *in,
00322                        uint64_t len
00323 );
00324
00331 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_256_update() instead.")
00332 void
00333 aes_gcm_enc_256_update(const struct isal_gcm_key_data *key_data,
00334                        struct isal_gcm_context_data *context_data,
00335                        uint8_t *out,
00336                        const uint8_t *in,
00337                        uint64_t len
00338 );
00339
00346 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_128_update() instead.")
00347 void
00348 aes_gcm_dec_128_update(const struct isal_gcm_key_data *key_data,
00349                        struct isal_gcm_context_data *context_data,
00350                        uint8_t *out,
00351                        const uint8_t *in,
00352                        uint64_t len
00353 );
00354
00361 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_256_update() instead.")
00362 void
00363 aes_gcm_dec_256_update(const struct isal_gcm_key_data *key_data,
00364                        struct isal_gcm_context_data *context_data,
00365                        uint8_t *out,
00366                        const uint8_t *in,
00367                        uint64_t len
00368 );
00369
00376 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_128_finalize() instead.")
00377 void
00378 aes_gcm_enc_128_finalize(const struct isal_gcm_key_data *key_data,
00379                          struct isal_gcm_context_data *context_data,
00380                          uint8_t *auth_tag,
00381                          uint64_t auth_tag_len
00384 );
00385
00392 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_256_finalize() instead.")
00393 void
00394 aes_gcm_enc_256_finalize(const struct isal_gcm_key_data *key_data,
00395                          struct isal_gcm_context_data *context_data,
00396                          uint8_t *auth_tag,
00397                          uint64_t auth_tag_len
00400 );
00401
00408 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_128_finalize() instead.")
00409 void
00410 aes_gcm_dec_128_finalize(const struct isal_gcm_key_data *key_data,
00411                          struct isal_gcm_context_data *context_data,
00412                          uint8_t *auth_tag,
00413                          uint64_t auth_tag_len
00416 );
00417
00424 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_256_finalize() instead.")
00425 void
00426 aes_gcm_dec_256_finalize(const struct isal_gcm_key_data *key_data,
00427                          struct isal_gcm_context_data *context_data,
00428                          uint8_t *auth_tag,
00429                          uint64_t auth_tag_len
00432 );
00433
00443 ISAL_DEPRECATED("Please use isal_aes_gcm_pre_128() instead.")
00444 void
00445 aes_gcm_pre_128(const void *key,
00446                 struct isal_gcm_key_data *key_data
```

```
00447 );
00448
00458 ISAL_DEPRECATED("Please use isal_aes_gcm_pre_256() instead.")
00459 void
00460 aes_gcm_pre_256(const void *key,
00461                 struct isal_gcm_key_data *key_data
00462 );
00463
00464 /* ---- NT versions ---- */
00475 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_128_nt() instead.")
00476 void
00477 aes_gcm_enc_128_nt(const struct isal_gcm_key_data *key_data,
00478                    struct isal_gcm_context_data *context_data,
00479                    uint8_t *out,
00480                    uint8_t const *in,
00481                    uint64_t len,
00482                    uint8_t *iv,
00484                    uint8_t const *aad,
00485                    uint64_t aad_len,
00486                    uint8_t *auth_tag,
00487                    uint64_t auth_tag_len
00490 );
00491
00502 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_256_nt() instead.")
00503 void
00504 aes_gcm_enc_256_nt(const struct isal_gcm_key_data *key_data,
00505                    struct isal_gcm_context_data *context_data,
00506                    uint8_t *out,
00507                    uint8_t const *in,
00508                    uint64_t len,
00509                    uint8_t *iv,
00511                    uint8_t const *aad,
00512                    uint64_t aad_len,
00513                    uint8_t *auth_tag,
00514                    uint64_t auth_tag_len
00517 );
00518
00529 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_128_nt() instead.")
00530 void
00531 aes_gcm_dec_128_nt(const struct isal_gcm_key_data *key_data,
00532                    struct isal_gcm_context_data *context_data,
00533                    uint8_t *out,
00534                    uint8_t const *in,
00535                    uint64_t len,
00536                    uint8_t *iv,
00538                    uint8_t const *aad,
00539                    uint64_t aad_len,
00540                    uint8_t *auth_tag,
00541                    uint64_t auth_tag_len
00544 );
00545
00556 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_256_nt() instead.")
00557 void
00558 aes_gcm_dec_256_nt(const struct isal_gcm_key_data *key_data,
00559                    struct isal_gcm_context_data *context_data,
00560                    uint8_t *out,
00561                    uint8_t const *in,
00562                    uint64_t len,
00563                    uint8_t *iv,
00565                    uint8_t const *aad,
00566                    uint64_t aad_len,
00567                    uint8_t *auth_tag,
00568                    uint64_t auth_tag_len
00571 );
00572
00585 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_128_update_nt() instead.")
00586 void
00587 aes_gcm_enc_128_update_nt(
00588         const struct isal_gcm_key_data *key_data,
00589         struct isal_gcm_context_data *context_data,
00590         uint8_t *out,
00591         const uint8_t *in,
00592         uint64_t len
00593 );
00594
00607 ISAL_DEPRECATED("Please use isal_aes_gcm_enc_256_update_nt() instead.")
00608 void
00609 aes_gcm_enc_256_update_nt(
00610         const struct isal_gcm_key_data *key_data,
00611         struct isal_gcm_context_data *context_data,
00612         uint8_t *out,
```

```
00613         const uint8_t *in,
00614         uint64_t len
00615 );
00616
00629 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_128_update_nt() instead.")
00630 void
00631 aes_gcm_dec_128_update_nt(
00632         const struct isal_gcm_key_data *key_data,
00633         struct isal_gcm_context_data *context_data,
00634         uint8_t *out,
00635         const uint8_t *in,
00636         uint64_t len
00637 );
00638
00651 ISAL_DEPRECATED("Please use isal_aes_gcm_dec_256_update_nt() instead.")
00652 void
00653 aes_gcm_dec_256_update_nt(
00654         const struct isal_gcm_key_data *key_data,
00655         struct isal_gcm_context_data *context_data,
00656         uint8_t *out,
00657         const uint8_t *in,
00658         uint64_t len
00659 );
00660
00669 int
00670 isal_aes_gcm_enc_128(
00671         const struct isal_gcm_key_data *key_data,
00672         struct isal_gcm_context_data *context_data,
00673         uint8_t *out,
00674         const uint8_t *in,
00675         const uint64_t len,
00676         const uint8_t *iv,
00678         const uint8_t *aad,
00679         const uint64_t aad_len,
00680         uint8_t *auth_tag,
00681         const uint64_t auth_tag_len
00684 );
00685
00694 int
00695 isal_aes_gcm_enc_256(
00696         const struct isal_gcm_key_data *key_data,
00697         struct isal_gcm_context_data *context_data,
00698         uint8_t *out,
00699         const uint8_t *in,
00700         const uint64_t len,
00701         const uint8_t *iv,
00703         const uint8_t *aad,
00704         const uint64_t aad_len,
00705         uint8_t *auth_tag,
00706         const uint64_t auth_tag_len
00709 );
00710
00719 int
00720 isal_aes_gcm_dec_128(
00721         const struct isal_gcm_key_data *key_data,
00722         struct isal_gcm_context_data *context_data,
00723         uint8_t *out,
00724         const uint8_t *in,
00725         const uint64_t len,
00726         const uint8_t *iv,
00728         const uint8_t *aad,
00729         const uint64_t aad_len,
00730         uint8_t *auth_tag,
00731         const uint64_t auth_tag_len
00734 );
00735
00744 int
00745 isal_aes_gcm_dec_256(
00746         const struct isal_gcm_key_data *key_data,
00747         struct isal_gcm_context_data *context_data,
00748         uint8_t *out,
00749         const uint8_t *in,
00750         const uint64_t len,
00751         const uint8_t *iv,
00753         const uint8_t *aad,
00754         const uint64_t aad_len,
00755         uint8_t *auth_tag,
00756         const uint64_t auth_tag_len
00759 );
00760
00769 int
```

```
00770 isal_aes_gcm_init_128(const struct isal_gcm_key_data *key_data,
00771                       struct isal_gcm_context_data *context_data,
00772                       const uint8_t *iv,
00774                       const uint8_t *aad,
00775                       const uint64_t aad_len
00776 );
00777
00786 int
00787 isal_aes_gcm_init_256(const struct isal_gcm_key_data *key_data,
00788                       struct isal_gcm_context_data *context_data,
00789                       const uint8_t *iv,
00791                       const uint8_t *aad,
00792                       const uint64_t aad_len
00793 );
00794
00803 int
00804 isal_aes_gcm_enc_128_update(
00805         const struct isal_gcm_key_data *key_data,
00806         struct isal_gcm_context_data *context_data,
00807         uint8_t *out,
00808         const uint8_t *in,
00809         const uint64_t len
00810 );
00811
00820 int
00821 isal_aes_gcm_enc_256_update(
00822         const struct isal_gcm_key_data *key_data,
00823         struct isal_gcm_context_data *context_data,
00824         uint8_t *out,
00825         const uint8_t *in,
00826         const uint64_t len
00827 );
00828
00837 int
00838 isal_aes_gcm_dec_128_update(
00839         const struct isal_gcm_key_data *key_data,
00840         struct isal_gcm_context_data *context_data,
00841         uint8_t *out,
00842         const uint8_t *in,
00843         const uint64_t len
00844 );
00845
00854 int
00855 isal_aes_gcm_dec_256_update(
00856         const struct isal_gcm_key_data *key_data,
00857         struct isal_gcm_context_data *context_data,
00858         uint8_t *out,
00859         const uint8_t *in,
00860         const uint64_t len
00861 );
00862
00871 int
00872 isal_aes_gcm_enc_128_finalize(
00873         const struct isal_gcm_key_data *key_data,
00874         struct isal_gcm_context_data *context_data,
00875         uint8_t *auth_tag,
00876         const uint64_t auth_tag_len
00879 );
00880
00889 int
00890 isal_aes_gcm_enc_256_finalize(
00891         const struct isal_gcm_key_data *key_data,
00892         struct isal_gcm_context_data *context_data,
00893         uint8_t *auth_tag,
00894         const uint64_t auth_tag_len
00897 );
00898
00907 int
00908 isal_aes_gcm_dec_128_finalize(
00909         const struct isal_gcm_key_data *key_data,
00910         struct isal_gcm_context_data *context_data,
00911         uint8_t *auth_tag,
00912         const uint64_t auth_tag_len
00915 );
00916
00925 int
00926 isal_aes_gcm_dec_256_finalize(
00927         const struct isal_gcm_key_data *key_data,
00928         struct isal_gcm_context_data *context_data,
00929         uint8_t *auth_tag,
00930         const uint64_t auth_tag_len
```

```
00933 );
00934
00947 int
00948 isal_aes_gcm_pre_128(const void *key,
00949                       struct isal_gcm_key_data *key_data
00950 );
00951
00964 int
00965 isal_aes_gcm_pre_256(const void *key,
00966                       struct isal_gcm_key_data *key_data
00967 );
00968
00969 /* ---- NT versions ---- */
00983 int
00984 isal_aes_gcm_enc_128_nt(
00985         const struct isal_gcm_key_data *key_data,
00986         struct isal_gcm_context_data *context_data,
00987         uint8_t *out,
00988         const uint8_t *in,
00989         const uint64_t len,
00990         const uint8_t *iv,
00992         const uint8_t *aad,
00993         const uint64_t aad_len,
00994         uint8_t *auth_tag,
00995         const uint64_t auth_tag_len
00998 );
00999
01013 int
01014 isal_aes_gcm_enc_256_nt(
01015         const struct isal_gcm_key_data *key_data,
01016         struct isal_gcm_context_data *context_data,
01017         uint8_t *out,
01018         const uint8_t *in,
01019         const uint64_t len,
01020         const uint8_t *iv,
01022         const uint8_t *aad,
01023         const uint64_t aad_len,
01024         uint8_t *auth_tag,
01025         const uint64_t auth_tag_len
01028 );
01029
01043 int
01044 isal_aes_gcm_dec_128_nt(
01045         const struct isal_gcm_key_data *key_data,
01046         struct isal_gcm_context_data *context_data,
01047         uint8_t *out,
01048         const uint8_t *in,
01049         const uint64_t len,
01050         const uint8_t *iv,
01052         const uint8_t *aad,
01053         const uint64_t aad_len,
01054         uint8_t *auth_tag,
01055         const uint64_t auth_tag_len
01058 );
01059
01073 int
01074 isal_aes_gcm_dec_256_nt(
01075         const struct isal_gcm_key_data *key_data,
01076         struct isal_gcm_context_data *context_data,
01077         uint8_t *out,
01078         const uint8_t *in,
01079         const uint64_t len,
01080         const uint8_t *iv,
01082         const uint8_t *aad,
01083         const uint64_t aad_len,
01084         uint8_t *auth_tag,
01085         const uint64_t auth_tag_len
01088 );
01089
01105 int
01106 isal_aes_gcm_enc_128_update_nt(
01107         const struct isal_gcm_key_data *key_data,
01108         struct isal_gcm_context_data *context_data,
01109         uint8_t *out,
01110         const uint8_t *in,
01111         const uint64_t len
01112 );
01113
01129 int
01130 isal_aes_gcm_enc_256_update_nt(
01131         const struct isal_gcm_key_data *key_data,
```

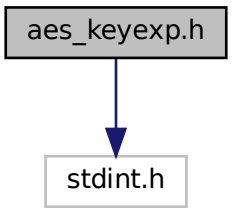
```
01132         struct isal_gcm_context_data *context_data,
01133         uint8_t *out,
01134         const uint8_t *in,
01135         const uint64_t len
01136 );
01137
01153 int
01154 isal_aes_gcm_dec_128_update_nt(
01155         const struct isal_gcm_key_data *key_data,
01156         struct isal_gcm_context_data *context_data,
01157         uint8_t *out,
01158         const uint8_t *in,
01159         const uint64_t len
01160 );
01161
01177 int
01178 isal_aes_gcm_dec_256_update_nt(
01179         const struct isal_gcm_key_data *key_data,
01180         struct isal_gcm_context_data *context_data,
01181         uint8_t *out,
01182         const uint8_t *in,
01183         const uint64_t len
01184 );
01185
01186 #ifdef __cplusplus
01187 }
01188 #endif //__cplusplus
01189 #endif // ifndef _AES_GCM_h
```

10.5 aes_keyexp.h File Reference

AES key expansion functions.

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for aes_keyexp.h:



Functions

- void aes_keyexp_128 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

  AES key expansion 128 bit.
- void aes_keyexp_192 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

  AES key expansion 192 bit.
- void aes_keyexp_256 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

AES key expansion 256 bit.

- int isal_aes_keyexp_128 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

  AES key expansion 128 bit.
- int isal_aes_keyexp_192 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

  AES key expansion 192 bit.
- int isal_aes_keyexp_256 (const uint8_t *key, uint8_t *exp_key_enc, uint8_t *exp_key_dec)

  AES key expansion 256 bit.

10.5.1 Detailed Description

AES key expansion functions.

This defines the interface to key expansion functions.

10.5.2 Function Documentation

10.5.2.1 aes_keyexp_128()

```
void aes_keyexp_128 (
            const uint8_t * key,
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 128 bit.

Requires SSE4.1

Deprecated Please use isal_aes_keyexp_128() instead.

Parameters

key	input key for AES-128, 16 bytes
exp_key_enc	expanded encryption keys, 16*11 bytes
exp_key_dec	expanded decryption keys, 16*11 bytes

10.5.2.2 aes_keyexp_192()

```
void aes_keyexp_192 (
            const uint8_t * key,
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 192 bit.

Requires SSE4.1

Deprecated Please use isal_aes_keyexp_192() instead.

Parameters

key	input key for AES-192, 16∗1.5 bytes
exp_key_enc	expanded encryption keys, 16∗13 bytes
exp_key_dec	expanded decryption keys, 16∗13 bytes

10.5.2.3 aes_keyexp_256()

```
void aes_keyexp_256 (
            const uint8_t * key,
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 256 bit.

Requires SSE4.1

Deprecated Please use isal_aes_keyexp_256() instead.

Parameters

key	input key for AES-256, 16∗2 bytes
exp_key_enc	expanded encryption keys, 16∗15 bytes
exp_key_dec	expanded decryption keys, 16∗15 bytes

10.5.2.4 isal_aes_keyexp_128()

```
int isal_aes_keyexp_128 (
            const uint8_t * key,
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 128 bit.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key	input key for AES-128, 16 bytes
exp_key_enc	expanded encryption keys, 16∗11 bytes
exp_key_dec	expanded decryption keys, 16∗11 bytes

10.5.2.5 isal_aes_keyexp_192()

```
int isal_aes_keyexp_192 (
            const uint8_t * key,
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 192 bit.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key	input key for AES-192, 24 bytes
exp_key_enc	expanded encryption keys, 16∗13 bytes
exp_key_dec	expanded decryption keys, 16∗13 bytes

10.5.2.6 isal_aes_keyexp_256()

```
int isal_aes_keyexp_256 (
            const uint8_t * key,
```

```
            uint8_t * exp_key_enc,
            uint8_t * exp_key_dec )
```

AES key expansion 256 bit.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

key	input key for AES-256, 32 bytes
exp_key_enc	expanded encryption keys, 16∗15 bytes
exp_key_dec	expanded decryption keys, 16∗15 bytes

10.6 aes_keyexp.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _KEYEXP_128_H
00031 #define _KEYEXP_128_H
```

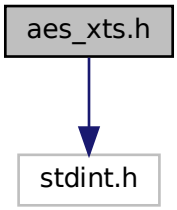
```
00032
00040 #include <stdint.h>
00041 #include "types.h"
00042
00043 #ifdef __cplusplus
00044 extern "C" {
00045 #endif
00046
00052 ISAL_DEPRECATED("Please use isal_aes_keyexp_128() instead")
00053 void
00054 aes_keyexp_128(const uint8_t *key,
00055                uint8_t *exp_key_enc,
00056                uint8_t *exp_key_dec
00057 );
00058
00064 ISAL_DEPRECATED("Please use isal_aes_keyexp_192() instead")
00065 void
00066 aes_keyexp_192(const uint8_t *key,
00067                uint8_t *exp_key_enc,
00068                uint8_t *exp_key_dec
00069 );
00070
00076 ISAL_DEPRECATED("Please use isal_aes_keyexp_256() instead")
00077 void
00078 aes_keyexp_256(const uint8_t *key,
00079                uint8_t *exp_key_enc,
00080                uint8_t *exp_key_dec
00081 );
00082
00090 int
00091 isal_aes_keyexp_128(const uint8_t *key,
00092                     uint8_t *exp_key_enc,
00093                     uint8_t *exp_key_dec
00094 );
00095
00103 int
00104 isal_aes_keyexp_192(const uint8_t *key,
00105                     uint8_t *exp_key_enc,
00106                     uint8_t *exp_key_dec
00107 );
00108
00116 int
00117 isal_aes_keyexp_256(const uint8_t *key,
00118                     uint8_t *exp_key_enc,
00119                     uint8_t *exp_key_dec
00120 );
00121
00122 #ifdef __cplusplus
00123 }
00124 #endif //__cplusplus
00125 #endif // ifndef _KEYEXP_128_H
```

10.7 aes_xts.h File Reference

AES XTS encryption function prototypes.

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for aes_xts.h:



Functions

- void XTS_AES_128_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

  XTS-AES-128 Encryption.
- void XTS_AES_128_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

  XTS-AES-128 Encryption with pre-expanded keys.
- void XTS_AES_128_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

  XTS-AES-128 Decryption.
- void XTS_AES_128_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

  XTS-AES-128 Decryption with pre-expanded keys.
- void XTS_AES_256_enc (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

  XTS-AES-256 Encryption.
- void XTS_AES_256_enc_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *pt, uint8_t *ct)

  XTS-AES-256 Encryption with pre-expanded keys.
- void XTS_AES_256_dec (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

  XTS-AES-256 Decryption.
- void XTS_AES_256_dec_expanded_key (uint8_t *k2, uint8_t *k1, uint8_t *TW_initial, uint64_t N, const uint8_t *ct, uint8_t *pt)

  XTS-AES-256 Decryption with pre-expanded keys.
- int isal_aes_xts_enc_128 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len↩_bytes, const void *in, void *out)

  XTS-AES-128 Encryption.
- int isal_aes_xts_enc_128_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

  XTS-AES-128 Encryption with pre-expanded keys.
- int isal_aes_xts_dec_128 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len↩_bytes, const void *in, void *out)

  XTS-AES-128 Decryption.
- int isal_aes_xts_dec_128_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

XTS-AES-128 Decryption with pre-expanded keys.

- int isal_aes_xts_enc_256 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

  XTS-AES-256 Encryption.
- int isal_aes_xts_enc_256_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

  XTS-AES-256 Encryption with pre-expanded keys.
- int isal_aes_xts_dec_256 (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

  XTS-AES-256 Decryption.
- int isal_aes_xts_dec_256_expanded_key (const uint8_t *k2, const uint8_t *k1, const uint8_t *initial_tweak, const uint64_t len_bytes, const void *in, void *out)

  XTS-AES-256 Decryption with pre-expanded keys.

10.7.1 Detailed Description

AES XTS encryption function prototypes.

This defines the interface to optimized AES XTS functions

Pre-expanded keys

For key encryption, pre-expanded keys are stored in the order that they will be used. As an example, if Key[0] is the 128-bit initial key used for an AES-128 encryption, the rest of the keys are stored as follows:

- Key[0] : Initial encryption key
- Key[1] : Round 1 encryption key
- Key[2] : Round 2 encryption key
- ...
- Key[10] : Round 10 encryption key

For decryption, the order of keys is reversed. However, we apply the necessary aesimc instructions before storing the expanded keys. For the same key used above, the pre-expanded keys will be stored as follows:

- Key[0] : Round 10 encryption key
- Key[1] : aesimc(Round 9 encryption key)
- Key[2] : aesimc(Round 8 encryption key)
- ...
- Key[9] : aesimc(Round 1 encryption key)
- Key[10] : Initial encryption key

Note: The expanded key decryption requires a decryption key only for the block decryption step. The tweak step in the expanded key decryption requires the same expanded encryption key that is used in the expanded key encryption.

Input and Output Buffers

The input and output buffers can be overlapping as long as the output buffer pointer is not less than the input buffer pointer. If the two pointers are the same, then encryption/decryption will occur in-place.

Data Length

- The functions support data length of any bytes greater than or equal to 16 bytes.
- Data length is a 64-bit value, which makes the largest possible data length 2^64 - 1 bytes.
- For data lengths from 0 to 15 bytes, the functions return without any error codes, without reading or writing any data.
- The functions only support byte lengths, not bits.

Initial Tweak

The functions accept a 128-bit initial tweak value. The user is responsible for padding the initial tweak value to this length.

Data Alignment

The input and output buffers, keys, pre-expanded keys and initial tweak value are not required to be aligned to 16 bytes, any alignment works.

10.7.2 Function Documentation

10.7.2.1 isal_aes_xts_dec_128()

```
int isal_aes_xts_dec_128 (
             const uint8_t * k2,
             const uint8_t * k1,
             const uint8_t * initial_tweak,
             const uint64_t len_bytes,
             const void * in,
             void * out )
```

XTS-AES-128 Decryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	key used for tweaking, 16 bytes
k1	key used for decryption of tweaked ciphertext, 16 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	ciphertext sector input data
out	plaintext sector output data

10.7.2.2 isal_aes_xts_dec_128_expanded_key()

```
int isal_aes_xts_dec_128_expanded_key (
             const uint8_t * k2,
             const uint8_t * k1,
             const uint8_t * initial_tweak,
             const uint64_t len_bytes,
             const void * in,
             void * out )
```

XTS-AES-128 Decryption with pre-expanded keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	expanded key used for tweaking, 16∗11 bytes
k1	expanded key used for decryption of tweaked ciphertext, 16∗11 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	ciphertext sector input data

Parameters

out	plaintext sector output data

10.7.2.3 isal_aes_xts_dec_256()

```
int isal_aes_xts_dec_256 (
            const uint8_t * k2,
            const uint8_t * k1,
            const uint8_t * initial_tweak,
            const uint64_t len_bytes,
            const void * in,
            void * out )
```

XTS-AES-256 Decryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	key used for tweaking, 16 bytes
k1	key used for decryption of tweaked ciphertext, 16∗2 bytes
initial_tweak	initial tweak value, 16∗2 bytes
len_bytes	sector size, in bytes
in	ciphertext sector input data
out	plaintext sector output data

10.7.2.4 isal_aes_xts_dec_256_expanded_key()

```
int isal_aes_xts_dec_256_expanded_key (
            const uint8_t * k2,
            const uint8_t * k1,
            const uint8_t * initial_tweak,
            const uint64_t len_bytes,
```

```
            const void * in,
            void * out )
```

XTS-AES-256 Decryption with pre-expanded keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	expanded key used for tweaking, 16∗15 bytes
k1	expanded key used for decryption of tweaked ciphertext, 16∗15 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	ciphertext sector input data
out	plaintext sector output data

10.7.2.5 isal_aes_xts_enc_128()

```
int isal_aes_xts_enc_128 (
            const uint8_t * k2,
            const uint8_t * k1,
            const uint8_t * initial_tweak,
            const uint64_t len_bytes,
            const void * in,
            void * out )
```

XTS-AES-128 Encryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	key used for tweaking, 16 bytes
k1	key used for encryption of tweaked plaintext, 16 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	plaintext sector input data
out	ciphertext sector output data

10.7.2.6 isal_aes_xts_enc_128_expanded_key()

```
int isal_aes_xts_enc_128_expanded_key (
             const uint8_t * k2,
             const uint8_t * k1,
             const uint8_t * initial_tweak,
             const uint64_t len_bytes,
             const void * in,
             void * out )
```

XTS-AES-128 Encryption with pre-expanded keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	expnaded key used for tweaking, 16∗11 bytes
k1	expanded key used for encryption of tweaked plaintext, 16∗11 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	plaintext sector input data

Parameters

out	ciphertext sector output data

10.7.2.7 isal_aes_xts_enc_256()

```
int isal_aes_xts_enc_256 (
            const uint8_t * k2,
            const uint8_t * k1,
            const uint8_t * initial_tweak,
            const uint64_t len_bytes,
            const void * in,
            void * out )
```

XTS-AES-256 Encryption.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	key used for tweaking, 16∗2 bytes
k1	key used for encryption of tweaked plaintext, 16∗2 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	plaintext sector input data
out	ciphertext sector output data

10.7.2.8 isal_aes_xts_enc_256_expanded_key()

```
int isal_aes_xts_enc_256_expanded_key (
            const uint8_t * k2,
            const uint8_t * k1,
            const uint8_t * initial_tweak,
            const uint64_t len_bytes,
```

```
            const void * in,
            void * out )
```

XTS-AES-256 Encryption with pre-expanded keys.

Requires AES extensions and SSE4.1 for x86 or ASIMD for ARM

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

Parameters

k2	expnaded key used for tweaking, 16∗15 bytes
k1	expanded key used for encryption of tweaked plaintext, 16∗15 bytes
initial_tweak	initial tweak value, 16 bytes
len_bytes	sector size, in bytes
in	plaintext sector input data
out	ciphertext sector output data

10.7.2.9 XTS_AES_128_dec()

```
void XTS_AES_128_dec (
            uint8_t * k2,
            uint8_t * k1,
            uint8_t * TW_initial,
            uint64_t N,
            const uint8_t * ct,
            uint8_t * pt )
```

XTS-AES-128 Decryption.

Requires AES-NI

Deprecated Please use isal_aes_xts_dec_128() instead.

Parameters

k2	key used for tweaking, 16 bytes

Parameters

k1	key used for decryption of tweaked ciphertext, 16 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
ct	ciphertext sector input data
pt	plaintext sector output data

10.7.2.10 XTS_AES_128_dec_expanded_key()

```
void XTS_AES_128_dec_expanded_key (
            uint8_t * k2,
            uint8_t * k1,
            uint8_t * TW_initial,
            uint64_t N,
            const uint8_t * ct,
            uint8_t * pt )
```

XTS-AES-128 Decryption with pre-expanded keys.

Requires AES-NI

Deprecated Please use isal_aes_xts_dec_128_expanded_key() instead.

Parameters

k2	expanded key used for tweaking, 16∗11 bytes - encryption key is used
k1	expanded decryption key used for decryption of tweaked ciphertext, 16∗11 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
ct	ciphertext sector input data
pt	plaintext sector output data

10.7.2.11 XTS_AES_128_enc()

```
void XTS_AES_128_enc (
            uint8_t * k2,
            uint8_t * k1,
            uint8_t * TW_initial,
            uint64_t N,
            const uint8_t * pt,
            uint8_t * ct )
```

XTS-AES-128 Encryption.

Requires AES-NI

Deprecated Please use isal_aes_xts_enc_128() instead.

Parameters

k2	key used for tweaking, 16 bytes
k1	key used for encryption of tweaked plaintext, 16 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
pt	plaintext sector input data
ct	ciphertext sector output data

10.7.2.12 XTS_AES_128_enc_expanded_key()

```
void XTS_AES_128_enc_expanded_key (
             uint8_t * k2,
             uint8_t * k1,
             uint8_t * TW_initial,
             uint64_t N,
             const uint8_t * pt,
             uint8_t * ct )
```

XTS-AES-128 Encryption with pre-expanded keys.

Requires AES-NI

Deprecated Please use isal_aes_xts_enc_128_expanded_key() instead.

Parameters

k2	expanded key used for tweaking, 16∗11 bytes
k1	expanded key used for encryption of tweaked plaintext, 16∗11 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
pt	plaintext sector input data
ct	ciphertext sector output data

10.7.2.13 XTS_AES_256_dec()

```
void XTS_AES_256_dec (
             uint8_t * k2,
```

```
            uint8_t * k1,
            uint8_t * TW_initial,
            uint64_t N,
            const uint8_t * ct,
            uint8_t * pt )
```

XTS-AES-256 Decryption.

Requires AES-NI

Deprecated Please use isal_aes_xts_dec_256() instead.

Parameters

k2	key used for tweaking, 16∗2 bytes
k1	key used for decryption of tweaked ciphertext, 16∗2 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
ct	ciphertext sector input data
pt	plaintext sector output data

10.7.2.14 XTS_AES_256_dec_expanded_key()

```
void XTS_AES_256_dec_expanded_key (
            uint8_t * k2,
            uint8_t * k1,
            uint8_t * TW_initial,
            uint64_t N,
            const uint8_t * ct,
            uint8_t * pt )
```

XTS-AES-256 Decryption with pre-expanded keys.

Requires AES-NI

Deprecated Please use isal_aes_xts_dec_256_expanded_key() instead.

Parameters

k2	expanded key used for tweaking, 16∗15 bytes - encryption key is used
k1	expanded decryption key used for decryption of tweaked ciphertext, 16∗15 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
ct	ciphertext sector input data
pt	plaintext sector output data

10.7.2.15 XTS_AES_256_enc()

```
void XTS_AES_256_enc (
             uint8_t * k2,
             uint8_t * k1,
             uint8_t * TW_initial,
             uint64_t N,
             const uint8_t * pt,
             uint8_t * ct )
```

XTS-AES-256 Encryption.

Requires AES-NI

Deprecated Please use isal_aes_xts_enc_256() instead.

Parameters

k2	key used for tweaking, 16∗2 bytes
k1	key used for encryption of tweaked plaintext, 16∗2 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
pt	plaintext sector input data
ct	ciphertext sector output data

10.7.2.16 XTS_AES_256_enc_expanded_key()

```
void XTS_AES_256_enc_expanded_key (
             uint8_t * k2,
             uint8_t * k1,
             uint8_t * TW_initial,
             uint64_t N,
             const uint8_t * pt,
             uint8_t * ct )
```

XTS-AES-256 Encryption with pre-expanded keys.

Requires AES-NI

Deprecated Please use isal_aes_xts_enc_256_expanded_key() instead.

Parameters

k2	expanded key used for tweaking, 16∗15 bytes

Parameters

k1	expanded key used for encryption of tweaked plaintext, 16∗15 bytes
TW_initial	initial tweak value, 16 bytes
N	sector size, in bytes
pt	plaintext sector input data
ct	ciphertext sector output data

10.8 aes_xts.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2024 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _AES_XTS_H
00031 #define _AES_XTS_H
00032
00099 #include <stdint.h>
00100 #include "types.h"
00101
00102 #ifdef __cplusplus
00103 extern "C" {
00104 #endif
00105
00106 /*
00107  * Define enums from API v2.24, so applications that were using this version
00108  * will still be compiled successfully.
00109  * This list does not need to be extended for new definitions.
00110  */
00111 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00112 /***** Previous hash constants and typedefs *****/
00113 #define AES_XTS_MIN_LEN ISAL_AES_XTS_MIN_LEN
00114 #define AES_XTS_MAX_LEN ISAL_AES_XTS_MAX_LEN
00115 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00116
00117 #define ISAL_AES_XTS_MIN_LEN 16
00118 #define ISAL_AES_XTS_MAX_LEN (1 « 24)
00119
00125 ISAL_DEPRECATED("Please use isal_aes_xts_enc_128() instead")
00126 void
00127 XTS_AES_128_enc(uint8_t *k2,
00128                 uint8_t *k1,
00129                 uint8_t *TW_initial,
```

```
00130                 uint64_t N,
00131                 const uint8_t *pt,
00132                 uint8_t *ct
00133 );
00134
00140 ISAL_DEPRECATED("Please use isal_aes_xts_enc_128_expanded_key() instead")
00141 void
00142 XTS_AES_128_enc_expanded_key(
00143         uint8_t *k2,
00144         uint8_t *k1,
00145         uint8_t *TW_initial,
00146         uint64_t N,
00147         const uint8_t *pt,
00148         uint8_t *ct
00149 );
00150
00156 ISAL_DEPRECATED("Please use isal_aes_xts_dec_128() instead")
00157 void
00158 XTS_AES_128_dec(uint8_t *k2,
00159                 uint8_t *k1,
00160                 uint8_t *TW_initial,
00161                 uint64_t N,
00162                 const uint8_t *ct,
00163                 uint8_t *pt
00164 );
00165
00171 ISAL_DEPRECATED("Please use isal_aes_xts_dec_128_expanded_key() instead")
00172 void
00173 XTS_AES_128_dec_expanded_key(
00174         uint8_t *k2,
00175         uint8_t *k1,
00177         uint8_t *TW_initial,
00178         uint64_t N,
00179         const uint8_t *ct,
00180         uint8_t *pt
00181 );
00182
00188 ISAL_DEPRECATED("Please use isal_aes_xts_enc_256() instead")
00189 void
00190 XTS_AES_256_enc(uint8_t *k2,
00191                 uint8_t *k1,
00192                 uint8_t *TW_initial,
00193                 uint64_t N,
00194                 const uint8_t *pt,
00195                 uint8_t *ct
00196 );
00197
00203 ISAL_DEPRECATED("Please use isal_aes_xts_enc_256_expanded_key() instead")
00204 void
00205 XTS_AES_256_enc_expanded_key(
00206         uint8_t *k2,
00207         uint8_t *k1,
00208         uint8_t *TW_initial,
00209         uint64_t N,
00210         const uint8_t *pt,
00211         uint8_t *ct
00212 );
00213
00219 ISAL_DEPRECATED("Please use isal_aes_xts_dec_256() instead")
00220 void
00221 XTS_AES_256_dec(uint8_t *k2,
00222                 uint8_t *k1,
00223                 uint8_t *TW_initial,
00224                 uint64_t N,
00225                 const uint8_t *ct,
00226                 uint8_t *pt
00227 );
00228
00234 ISAL_DEPRECATED("Please use isal_aes_xts_dec_256_expanded_key() instead")
00235 void
00236 XTS_AES_256_dec_expanded_key(
00237         uint8_t *k2,
00238         uint8_t *k1,
00240         uint8_t *TW_initial,
00241         uint64_t N,
00242         const uint8_t *ct,
00243         uint8_t *pt
00244 );
00245
00254 int
00255 isal_aes_xts_enc_128(const uint8_t *k2,
```

```
00256                     const uint8_t *k1,
00257                     const uint8_t *initial_tweak,
00258                     const uint64_t len_bytes,
00259                     const void *in,
00260                     void *out
00261 );
00262
00271 int
00272 isal_aes_xts_enc_128_expanded_key(
00273         const uint8_t *k2,
00274         const uint8_t *k1,
00275         const uint8_t *initial_tweak,
00276         const uint64_t len_bytes,
00277         const void *in,
00278         void *out
00279 );
00280
00289 int
00290 isal_aes_xts_dec_128(
00291         const uint8_t *k2,
00292         const uint8_t *k1,
00293         const uint8_t *initial_tweak,
00294         const uint64_t len_bytes,
00295         const void *in,
00296         void *out
00297 );
00298
00307 int
00308 isal_aes_xts_dec_128_expanded_key(
00309         const uint8_t *k2,
00310         const uint8_t *k1,
00311         const uint8_t *initial_tweak,
00312         const uint64_t len_bytes,
00313         const void *in,
00314         void *out
00315 );
00316
00325 int
00326 isal_aes_xts_enc_256(
00327         const uint8_t *k2,
00328         const uint8_t *k1,
00329         const uint8_t *initial_tweak,
00330         const uint64_t len_bytes,
00331         const void *in,
00332         void *out
00333 );
00334
00343 int
00344 isal_aes_xts_enc_256_expanded_key(
00345         const uint8_t *k2,
00346         const uint8_t *k1,
00347         const uint8_t *initial_tweak,
00348         const uint64_t len_bytes,
00349         const void *in,
00350         void *out
00351 );
00352
00361 int
00362 isal_aes_xts_dec_256(
00363         const uint8_t *k2,
00364         const uint8_t *k1,
00365         const uint8_t *initial_tweak,
00366         const uint64_t len_bytes,
00367         const void *in,
00368         void *out
00369 );
00370
00379 int
00380 isal_aes_xts_dec_256_expanded_key(
00381         const uint8_t *k2,
00382         const uint8_t *k1,
00383         const uint8_t *initial_tweak,
00384         const uint64_t len_bytes,
00385         const void *in,
00386         void *out
00387 );
00388 #ifdef __cplusplus
00389 }
00390 #endif
00391
00392 #endif //_AES_XTS_H
```

10.9 isal_crypto_api.h

```
00001 /**********************************************************************
00002   Copyright(c) 2024 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _ISAL_CRYPTO_API_H
00031 #define _ISAL_CRYPTO_API_H
00032
00033 #ifdef __cplusplus
00034 extern "C" {
00035 #endif
00036
00037 /* Utility macros */
00038 #define CONCAT_VERSION_(a, b, c) a##.##b##.##c
00039 #define CONCAT_VERSION(a, b, c)  CONCAT_VERSION_(a, b, c)
00040 #define TO_STRING_(a)            #a
00041 #define TO_STRING(a)             TO_STRING_(a)
00042
00043 /* Library version numbers */
00044 #define ISAL_CRYPTO_MAJOR_VERSION 2
00045 #define ISAL_CRYPTO_MINOR_VERSION 25
00046 #define ISAL_CRYPTO_PATCH_VERSION 0
00047
00048 #define ISAL_CRYPTO_MAKE_VERSION(maj, min, patch) ((maj) * 0x10000 + (min) * 0x100 + (patch))
00049 #define ISAL_CRYPTO_VERSION                                                          \
00050         ISAL_CRYPTO_MAKE_VERSION(ISAL_CRYPTO_MAJOR_VERSION, ISAL_CRYPTO_MINOR_VERSION,          \
00051                                  ISAL_CRYPTO_PATCH_VERSION)
00052 #define ISAL_CRYPTO_VERSION_STR                                                      \
00053         TO_STRING(CONCAT_VERSION(ISAL_CRYPTO_MAJOR_VERSION, ISAL_CRYPTO_MINOR_VERSION,          \
00054                                  ISAL_CRYPTO_PATCH_VERSION))
00055
00059 typedef enum {
00060         ISAL_CRYPTO_ERR_NONE = 0,
00061         ISAL_CRYPTO_ERR_NULL_SRC = 2000,
00062         ISAL_CRYPTO_ERR_NULL_DST,
00063         ISAL_CRYPTO_ERR_NULL_CTX,
00064         ISAL_CRYPTO_ERR_NULL_MGR,
00065         ISAL_CRYPTO_ERR_NULL_KEY,
00066         ISAL_CRYPTO_ERR_NULL_EXP_KEY,
00067         ISAL_CRYPTO_ERR_NULL_IV,
00068         ISAL_CRYPTO_ERR_NULL_AUTH,
00069         ISAL_CRYPTO_ERR_NULL_AAD,
00070         ISAL_CRYPTO_ERR_CIPH_LEN,
00071         ISAL_CRYPTO_ERR_AUTH_TAG_LEN,
00072         ISAL_CRYPTO_ERR_INVALID_FLAGS,
00073         ISAL_CRYPTO_ERR_ALREADY_PROCESSING,
00074         ISAL_CRYPTO_ERR_ALREADY_COMPLETED,
00075         ISAL_CRYPTO_ERR_XTS_NULL_TWEAK,
00076         ISAL_CRYPTO_ERR_XTS_SAME_KEYS,
00077         ISAL_CRYPTO_ERR_SELF_TEST,
00078         ISAL_CRYPTO_ERR_FIPS_INVALID_ALGO,
00079         ISAL_CRYPTO_ERR_WINDOW_SIZE,
00080         ISAL_CRYPTO_ERR_NULL_OFFSET,
00081         ISAL_CRYPTO_ERR_NULL_MATCH,
```

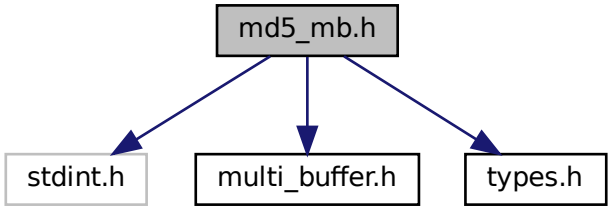
```
00082         ISAL_CRYPTO_ERR_NULL_MASK,
00083         ISAL_CRYPTO_ERR_NULL_INIT_VAL,
00084         ISAL_CRYPTO_ERR_FIPS_DISABLED,
00085         /* add new error types above this comment */
00086         ISAL_CRYPTO_ERR_MAX /* don't move this one */
00087 } ISAL_CRYPTO_ERROR;
00088
00108 int
00109 isal_self_tests(void);
00110
00116 const char *
00117 isal_crypto_get_version_str(void);
00118
00127 unsigned
00128 isal_crypto_get_version(void);
00129
00130 #ifdef __cplusplus
00131 }
00132 #endif //__cplusplus
00133 #endif // ifndef _ISAL_CRYPTO_API_H
```

10.10 md5_mb.h File Reference

Multi-buffer CTX API MD5 function prototypes and structures.

```
#include <stdint.h>
#include <string.h>
#include "multi_buffer.h"
#include "types.h"
```

Include dependency graph for md5_mb.h:



Data Structures

- struct ISAL_MD5_JOB

  Scheduler layer - Holds info describing a single MD5 job for the multi-buffer manager.
- struct ISAL_MD5_MB_ARGS_X32

  Scheduler layer - Holds arguments for submitted MD5 job.
- struct ISAL_MD5_LANE_DATA

  Scheduler layer - Lane data.
- struct ISAL_MD5_MB_JOB_MGR

Scheduler layer - Holds state for multi-buffer MD5 jobs.

- struct ISAL_MD5_HASH_CTX_MGR

  Context layer - Holds state for multi-buffer MD5 jobs.

- struct ISAL_MD5_HASH_CTX

  Context layer - Holds info describing a single MD5 job for the multi-buffer CTX manager.

Functions

- void md5_ctx_mgr_init (ISAL_MD5_HASH_CTX_MGR *mgr)

  Initialize the MD5 multi-buffer manager structure.

- ISAL_MD5_HASH_CTX * md5_ctx_mgr_submit (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

  Submit a new MD5 job to the multi-buffer manager.

- ISAL_MD5_HASH_CTX * md5_ctx_mgr_flush (ISAL_MD5_HASH_CTX_MGR *mgr)

  Finish all submitted MD5 jobs and return when complete.

- int isal_md5_ctx_mgr_init (ISAL_MD5_HASH_CTX_MGR *mgr)

  Initialize the MD5 multi-buffer manager structure.

- int isal_md5_ctx_mgr_submit (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx_in, ISAL_MD5_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

  Submit a new MD5 job to the multi-buffer manager.

- int isal_md5_ctx_mgr_flush (ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX **ctx_out)

  Finish all submitted MD5 jobs and return when complete.

10.10.1 Detailed Description

Multi-buffer CTX API MD5 function prototypes and structures.

Warning

: MD5 is considered unsafe, so it is recommended to use SHA256 instead.

Interface for multi-buffer MD5 functions

Multi-buffer MD5 Entire or First-Update..Update-Last

The interface to this multi-buffer hashing code is carried out through the context-level (CTX) init, submit and flush functions and the ISAL_MD5_HASH_CTX_MGR and ISAL_MD5_HASH_CTX objects. Numerous ISAL_MD5_HASH_CTX objects may be instantiated by the application for use with a single ISAL_MD5_HASH_CTX_MGR.

The CTX interface functions carry out the initialization and padding of the jobs entered by the user and add them to the multi-buffer manager. The lower level "scheduler" layer then processes the jobs in an out-of-order manner. The scheduler layer functions are internal and are not intended to be invoked directly. Jobs can be submitted to a CTX as a complete buffer to be hashed, using the ISAL_HASH_ENTIRE flag, or as partial jobs which can be started using the ISAL_HASH_FIRST flag, and later resumed or finished using the ISAL_HASH_UPDATE and ISAL_HASH_LAST flags respectively.

Note: The submit function does not require data buffers to be block sized.

The MD5 CTX interface functions are available for 4 architectures: SSE, AVX, AVX2 and AVX512. In addition, a multibinary interface is provided, which selects the appropriate architecture-specific function at runtime.

Usage: The application creates a ISAL_MD5_HASH_CTX_MGR object and initializes it with a call to md5_ctx_mgr↩ _init∗() function, where henceforth "∗" stands for the relevant suffix for each architecture; _sse, _avx, _avx2, _avx512 (or no suffix for the multibinary version). The ISAL_MD5_HASH_CTX_MGR object will be used to schedule processor resources, with up to 8 ISAL_MD5_HASH_CTX objects (or 16 in AVX2 case, 32 in AVX512 case) being processed at a time.

Each ISAL_MD5_HASH_CTX must be initialized before first use by the isal_hash_ctx_init macro defined in multi_buffer.h. After initialization, the application may begin computing a hash by giving the ISAL_MD5_HASH_CTX to a ISAL_MD5_HASH_CTX_MGR using the submit functions md5_ctx_mgr_submit∗() with the ISAL_HASH_FIRST flag set. When the ISAL_MD5_HASH_CTX is returned to the application (via this or a later call to md5_ctx_mgr_submit∗() or md5_ctx_mgr_flush∗()), the application can then re-submit it with another call to md5_ctx_mgr_submit∗(), but without the ISAL_HASH_FIRST flag set.

Ideally, on the last buffer for that hash, md5_ctx_mgr_submit_sse is called with ISAL_HASH_LAST, although it is also possible to submit the hash with ISAL_HASH_LAST and a zero length if necessary. When a ISAL_MD5_HASH_CTX is returned after having been submitted with ISAL_HASH_LAST, it will contain a valid hash. The ISAL_MD5_HASH_CTX can be reused immediately by submitting with ISAL_HASH_FIRST.

For example, you would submit hashes with the following flags for the following numbers of buffers:

- one buffer: ISAL_HASH_FIRST | ISAL_HASH_LAST (or, equivalently, ISAL_HASH_ENTIRE)
- two buffers: ISAL_HASH_FIRST, ISAL_HASH_LAST
- three buffers: ISAL_HASH_FIRST, ISAL_HASH_UPDATE, ISAL_HASH_LAST etc.

The order in which MD5_CTX objects are returned is in general different from the order in which they are submitted.

A few possible error conditions exist:

- Submitting flags other than the allowed entire/first/update/last values
- Submitting a context that is currently being managed by a ISAL_MD5_HASH_CTX_MGR.
- Submitting a context after ISAL_HASH_LAST is used but before ISAL_HASH_FIRST is set.

These error conditions are reported by returning the ISAL_MD5_HASH_CTX immediately after a submit with its error member set to a non-zero error code (defined in multi_buffer.h). No changes are made to the ISAL_MD5_HASH_CTX_MGR in the case of an error; no processing is done for other hashes.

10.10.2 Function Documentation

10.10.2.1 isal_md5_ctx_mgr_flush()

```
int isal_md5_ctx_mgr_flush (
            ISAL_MD5_HASH_CTX_MGR * mgr,
            ISAL_MD5_HASH_CTX ** ctx_out )
```

Finish all submitted MD5 jobs and return when complete.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs complete.

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.10.2.2 isal_md5_ctx_mgr_init()

```
int isal_md5_ctx_mgr_init (
            ISAL_MD5_HASH_CTX_MGR * mgr )
```

Initialize the MD5 multi-buffer manager structure.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.10.2.3 isal_md5_ctx_mgr_submit()

```
int isal_md5_ctx_mgr_submit (
            ISAL_MD5_HASH_CTX_MGR * mgr,
```

```
            ISAL_MD5_HASH_CTX * ctx_in,
            ISAL_MD5_HASH_CTX ** ctx_out,
            const void * buffer,
            const uint32_t len,
            const ISAL_HASH_CTX_FLAG flags )
```

Submit a new MD5 job to the multi-buffer manager.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
in	*ctx_in*	Pointer to structure holding input job ctx info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.
in	*buffer*	Pointer to buffer to be processed
in	*len*	Length of buffer (in bytes) to be processed
in	*flags*	Input flag specifying job type (first, update, last or entire)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.10.2.4 md5_ctx_mgr_flush()

```
ISAL_MD5_HASH_CTX * md5_ctx_mgr_flush (
            ISAL_MD5_HASH_CTX_MGR * mgr )
```

Finish all submitted MD5 jobs and return when complete.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_md5_ctx_mgr_submit() instead.

Parameters

mgr	Structure holding context level state info

Returns

NULL if no jobs to complete or pointer to jobs structure.

10.10.2.5 md5_ctx_mgr_init()

```
void md5_ctx_mgr_init (
             ISAL_MD5_HASH_CTX_MGR * mgr )
```

Initialize the MD5 multi-buffer manager structure.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_md5_ctx_mgr_init() instead.

Parameters

mgr	Structure holding context level state info

Returns

void

10.10.2.6 md5_ctx_mgr_submit()

```
ISAL_MD5_HASH_CTX * md5_ctx_mgr_submit (
             ISAL_MD5_HASH_CTX_MGR * mgr,
             ISAL_MD5_HASH_CTX * ctx,
             const void * buffer,
             uint32_t len,
             ISAL_HASH_CTX_FLAG flags )
```

Submit a new MD5 job to the multi-buffer manager.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_md5_ctx_mgr_submit() instead.

Parameters

mgr	Structure holding context level state info
ctx	Structure holding ctx job info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed
flags	Input flag specifying job type (first, update, last or entire)

Returns

NULL if no jobs complete or pointer to jobs structure.

10.11 md5_mb.h

Go to the documentation of this file.

```
00001 /*******************************************************************
00002   Copyright(c) 2011-2024 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 *******************************************************************/
00029
00030 #ifndef _MD5_MB_H_
00031 #define _MD5_MB_H_
00032
00108 #include <stdint.h>
00109 #include <string.h>
00110 #include "multi_buffer.h"
00111 #include "types.h"
00112
00113 #ifdef __cplusplus
00114 extern "C" {
00115 #endif
00116
00117 /*
00118  * Define enums from API v2.24, so applications that were using this version
00119  * will still be compiled successfully.
00120  * This list does not need to be extended for new definitions.
00121  */
00122 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00123 /***** Previous hash constants and typedefs *****/
00124 #define MD5_DIGEST_NWORDS       ISAL_MD5_DIGEST_NWORDS
00125 #define MD5_MAX_LANES           ISAL_MD5_MAX_LANES
00126 #define MD5_MIN_LANES           ISAL_MD5_MIN_LANES
00127 #define MD5_BLOCK_SIZE          ISAL_MD5_BLOCK_SIZE
00128 #define MD5_PADLENGTHFIELD_SIZE ISAL_MD5_PADLENGTHFIELD_SIZE
00129
00130 #define MD5_HASH_CTX     ISAL_MD5_HASH_CTX
00131 #define md5_digest_array isal_md5_digest_array
00132
00133 #define MD5_HASH_CTX_MGR ISAL_MD5_HASH_CTX_MGR
00134 #define MD5_JOB          ISAL_MD5_JOB
00135 #define MD5_WORD_T       ISAL_MD5_WORD_T
00136 #define MD5_MB_ARGS_X32  ISAL_MD5_MB_ARGS_X32
00137 #define MD5_LANE_DATA    ISAL_MD5_LANE_DATA
00138 #define MD5_MB_JOB_MGR   ISAL_MD5_MB_JOB_MGR
00139 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00140
00141 // Hash Constants and Typedefs
00142 #define ISAL_MD5_DIGEST_NWORDS       4
00143 #define ISAL_MD5_MAX_LANES           32
```

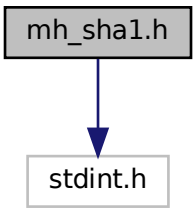
```
00144 #define ISAL_MD5_MIN_LANES           8
00145 #define ISAL_MD5_BLOCK_SIZE          64
00146 #define ISAL_MD5_PADLENGTHFIELD_SIZE 8
00147
00148 typedef uint32_t isal_md5_digest_array[ISAL_MD5_DIGEST_NWORDS][ISAL_MD5_MAX_LANES];
00149 typedef uint32_t ISAL_MD5_WORD_T;
00150
00153 typedef struct {
00154         uint8_t *buffer;
00155         uint32_t len;
00156         DECLARE_ALIGNED(uint32_t result_digest[ISAL_MD5_DIGEST_NWORDS], 64);
00158         ISAL_JOB_STS status;
00159         void *user_data;
00160 } ISAL_MD5_JOB;
00161
00164 typedef struct {
00165         isal_md5_digest_array digest;
00166         uint8_t *data_ptr[ISAL_MD5_MAX_LANES];
00167 } ISAL_MD5_MB_ARGS_X32;
00168
00171 typedef struct {
00172         ISAL_MD5_JOB *job_in_lane;
00173 } ISAL_MD5_LANE_DATA;
00174
00177 typedef struct {
00178         ISAL_MD5_MB_ARGS_X32 args;
00179         uint32_t lens[ISAL_MD5_MAX_LANES];
00180         uint64_t unused_lanes[4];
00181         ISAL_MD5_LANE_DATA ldata[ISAL_MD5_MAX_LANES];
00182         uint32_t num_lanes_inuse;
00183 } ISAL_MD5_MB_JOB_MGR;
00184
00187 typedef struct {
00188         ISAL_MD5_MB_JOB_MGR mgr;
00189 } ISAL_MD5_HASH_CTX_MGR;
00190
00194 typedef struct {
00195         ISAL_MD5_JOB job;               // Must be at struct offset 0.
00196         ISAL_HASH_CTX_STS status;
00197         ISAL_HASH_CTX_ERROR error;
00198         uint64_t total_length;
00199         const void *incoming_buffer;
00200         uint32_t incoming_buffer_length;
00201         uint8_t partial_block_buffer[ISAL_MD5_BLOCK_SIZE * 2];
00202         uint32_t partial_block_buffer_length;
00203         void *user_data;
00204 } ISAL_MD5_HASH_CTX;
00205
00206 /******************** multibinary function prototypes **********************/
00207
00216 ISAL_DEPRECATED("Please use isal_md5_ctx_mgr_init() instead")
00217 void
00218 md5_ctx_mgr_init(ISAL_MD5_HASH_CTX_MGR *mgr);
00219
00232 ISAL_DEPRECATED("Please use isal_md5_ctx_mgr_submit() instead")
00233 ISAL_MD5_HASH_CTX *
00234 md5_ctx_mgr_submit(ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx, const void *buffer,
00235                    uint32_t len, ISAL_HASH_CTX_FLAG flags);
00236
00245 ISAL_DEPRECATED("Please use isal_md5_ctx_mgr_submit() instead")
00246 ISAL_MD5_HASH_CTX *
00247 md5_ctx_mgr_flush(ISAL_MD5_HASH_CTX_MGR *mgr);
00248
00258 int
00259 isal_md5_ctx_mgr_init(ISAL_MD5_HASH_CTX_MGR *mgr);
00260
00277 int
00278 isal_md5_ctx_mgr_submit(ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX *ctx_in,
00279                         ISAL_MD5_HASH_CTX **ctx_out, const void *buffer, const uint32_t len,
00280                         const ISAL_HASH_CTX_FLAG flags);
00281
00294 int
00295 isal_md5_ctx_mgr_flush(ISAL_MD5_HASH_CTX_MGR *mgr, ISAL_MD5_HASH_CTX **ctx_out);
00296 #ifdef __cplusplus
00297 }
00298 #endif
00299
00300 #endif // _MD5_MB_H_
```

10.12 mh_sha1.h File Reference

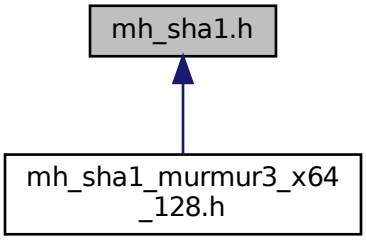
mh_sha1 function prototypes and structures

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for mh_sha1.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct isal_mh_sha1_ctx

  Holds info describing a single mh_sha1.

Enumerations

- enum isal_mh_sha1_ctx_error { ISAL_MH_SHA1_CTX_ERROR_NONE = 0 , ISAL_MH_SHA1_CTX_ERROR_NULL = -1 }

  CTX error flags.

Functions

- int mh_sha1_init (struct isal_mh_sha1_ctx *ctx)

  Initialize the isal_mh_sha1_ctx structure.
- int mh_sha1_update (struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha1 update.
- int mh_sha1_finalize (struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest)

  Finalize the message digests for multi-hash sha1.
- int mh_sha1_update_base (struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha1 update.
- int mh_sha1_finalize_base (struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest)

  Finalize the message digests for multi-hash sha1.
- int isal_mh_sha1_init (struct isal_mh_sha1_ctx *ctx)

  Initialize the isal_mh_sha1_ctx structure.
- int isal_mh_sha1_update (struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha1 update.
- int isal_mh_sha1_finalize (struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest)

  Finalize the message digests for multi-hash sha1.

10.12.1 Detailed Description

mh_sha1 function prototypes and structures

Interface for mh_sha1 functions

mh_sha1 Init-Update..Update-Finalize

This file defines the interface to optimized functions used in mh_sha1. The definition of multi-hash SHA1(mh_sha1, for short) is: Pad the buffer in SHA1 style until the total length is a multiple of 4*16*16 (words-width * parallel-segments * block-size); Hash the buffer in parallel, generating digests of 4*16*5 (words-width*parallel-segments* digest-size); Treat the set of digests as another data buffer, and generate a final SHA1 digest for it.

Example

```
uint32_t mh_sha1_digest[ISAL_SHA1_DIGEST_WORDS];
struct isal_mh_sha1_ctx *ctx;

ctx = malloc(sizeof(struct isal_mh_sha1_ctx));
isal_mh_sha1_init(ctx);
isal_mh_sha1_update(ctx, buff, block_len);
isal_mh_sha1_finalize(ctx, mh_sha1_digest);
```

10.12.2 Enumeration Type Documentation

10.12.2.1 isal_mh_sha1_ctx_error

```
enum isal_mh_sha1_ctx_error
```

CTX error flags.

Enumerator

ISAL_MH_SHA1_CTX_ERROR_NONE	ISAL_MH_SHA1_CTX_ERROR_NONE.
ISAL_MH_SHA1_CTX_ERROR_NULL	ISAL_MH_SHA1_CTX_ERROR_NULL.

10.12.3 Function Documentation

10.12.3.1 isal_mh_sha1_finalize()

```
int isal_mh_sha1_finalize (
            struct isal_mh_sha1_ctx * ctx,
            void * mh_sha1_digest )
```

Finalize the message digests for multi-hash sha1.

Place the message digest in mh_sha1_digest which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1 info
mh_sha1_digest	The digest of mh_sha1

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.12.3.2 isal_mh_sha1_init()

```
int isal_mh_sha1_init (
            struct isal_mh_sha1_ctx * ctx )
```

Initialize the isal_mh_sha1_ctx structure.

Parameters

ctx	Structure holding mh_sha1 info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.12.3.3 isal_mh_sha1_update()

```
int isal_mh_sha1_update (
            struct isal_mh_sha1_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha1 update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.12.3.4 mh_sha1_finalize()

```
int mh_sha1_finalize (
            struct isal_mh_sha1_ctx * ctx,
            void * mh_sha1_digest )
```

Finalize the message digests for multi-hash sha1.

Place the message digest in mh_sha1_digest which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1 info
mh_sha1_digest	The digest of mh_sha1

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_finalize() instead.

10.12.3.5 mh_sha1_finalize_base()

```
int mh_sha1_finalize_base (
            struct isal_mh_sha1_ctx * ctx,
            void * mh_sha1_digest )
```

Finalize the message digests for multi-hash sha1.

Place the message digests in mh_sha1_digest, which must have enough space for the outputs. Base Finalize() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha1 info
mh_sha1_digest	The digest of mh_sha1

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_finalize() instead.

10.12.3.6 mh_sha1_init()

```
int mh_sha1_init (
            struct isal_mh_sha1_ctx * ctx )
```

Initialize the isal_mh_sha1_ctx structure.

Parameters

ctx	Structure holding mh_sha1 info

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_init() instead.

10.12.3.7 mh_sha1_update()

```
int mh_sha1_update (
            struct isal_mh_sha1_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha1 update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_update() instead.

10.12.3.8 mh_sha1_update_base()

```
int mh_sha1_update_base (
            struct isal_mh_sha1_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha1 update.

Can be called repeatedly to update hashes with new input data. Base update() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha1 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_update() instead.

10.13 mh_sha1.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _MH_SHA1_H_
00031 #define _MH_SHA1_H_
00032
00062 #include <stdint.h>
00063 #include "types.h"
00064
00065 #ifdef __cplusplus
00066 extern "C" {
00067 #endif
00068
00069 /*
00070  * Define enums from API v2.24, so applications that were using this version
00071  * will still be compiled successfully.
00072  * This list does not need to be extended for new definitions.
00073  */
00074 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00075 /***** Previous hash constants and typedefs *****/
00076 #define HASH_SEGS           ISAL_HASH_SEGS
00077 #define SHA1_BLOCK_SIZE     ISAL_SHA1_BLOCK_SIZE
00078 #define MH_SHA1_BLOCK_SIZE  ISAL_MH_SHA1_BLOCK_SIZE
00079 #define SHA1_DIGEST_WORDS   ISAL_SHA1_DIGEST_WORDS
00080 #define AVX512_ALIGNED      ISAL_AVX512_ALIGNED
00081
00082 #define MH_SHA1_CTX_ERROR_NONE ISAL_MH_SHA1_CTX_ERROR_NONE
00083 #define MH_SHA1_CTX_ERROR_NULL ISAL_MH_SHA1_CTX_ERROR_NULL
00084
00085 #define mh_sha1_ctx isal_mh_sha1_ctx
00086 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00087
00088 // External Interface Definition
00089 #define ISAL_HASH_SEGS          16
00090 #define ISAL_SHA1_BLOCK_SIZE    64
00091 #define ISAL_MH_SHA1_BLOCK_SIZE (ISAL_HASH_SEGS * ISAL_SHA1_BLOCK_SIZE)
00092 #define ISAL_SHA1_DIGEST_WORDS  5
00093 #define ISAL_AVX512_ALIGNED     64
```

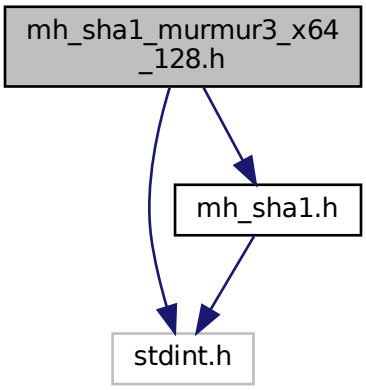
```
00094
00100 struct isal_mh_sha1_ctx {
00101         uint32_t mh_sha1_digest[ISAL_SHA1_DIGEST_WORDS];
00102
00103         uint64_t total_length;
00105         uint8_t partial_block_buffer[ISAL_MH_SHA1_BLOCK_SIZE * 2];
00107         uint8_t mh_sha1_interim_digests[sizeof(uint32_t) * ISAL_SHA1_DIGEST_WORDS * ISAL_HASH_SEGS];
00110         uint8_t frame_buffer[ISAL_MH_SHA1_BLOCK_SIZE + ISAL_AVX512_ALIGNED];
00113 };
00114
00119 enum isal_mh_sha1_ctx_error {
00120         ISAL_MH_SHA1_CTX_ERROR_NONE = 0,
00121         ISAL_MH_SHA1_CTX_ERROR_NULL = -1,
00122 };
00123
00124 /*****************************************************************
00125  * mh_sha1 API function prototypes
00126  *****************************************************************/
00127
00135 ISAL_DEPRECATED("Please use isal_mh_sha1_init() instead")
00136 int
00137 mh_sha1_init(struct isal_mh_sha1_ctx *ctx);
00138
00152 ISAL_DEPRECATED("Please use isal_mh_sha1_update() instead")
00153 int
00154 mh_sha1_update(struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len);
00155
00169 ISAL_DEPRECATED("Please use isal_mh_sha1_finalize() instead")
00170 int
00171 mh_sha1_finalize(struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest);
00172
00185 int
00186 mh_sha1_update_base(struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len);
00187
00200 int
00201 mh_sha1_finalize_base(struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest);
00202
00211 int
00212 isal_mh_sha1_init(struct isal_mh_sha1_ctx *ctx);
00213
00228 int
00229 isal_mh_sha1_update(struct isal_mh_sha1_ctx *ctx, const void *buffer, uint32_t len);
00230
00245 int
00246 isal_mh_sha1_finalize(struct isal_mh_sha1_ctx *ctx, void *mh_sha1_digest);
00247
00248 #ifdef __cplusplus
00249 }
00250 #endif
00251
00252 #endif
```

10.14 mh_sha1_murmur3_x64_128.h File Reference

mh_sha1_murmur3_x64_128 function prototypes and structures

```
#include <stdint.h>
#include "mh_sha1.h"
```

Include dependency graph for mh_sha1_murmur3_x64_128.h:



Data Structures

- struct isal_mh_sha1_murmur3_x64_128_ctx

  Holds info describing a single mh_sha1_murmur3_x64_128.

Enumerations

- enum isal_mh_sha1_murmur3_ctx_error { ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NONE = 0 , ISAL_MH_SHA1_MURMUR3_
  = -1 }

  CTX error flags.

Functions

- int mh_sha1_murmur3_x64_128_init (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, uint64_t murmur_seed)

  Initialize the isal_mh_sha1_murmur3_x64_128_ctx structure.
- int mh_sha1_murmur3_x64_128_update (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer, uint32_t len)

  Combined multi-hash and murmur hash update.
- int mh_sha1_murmur3_x64_128_finalize (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_sha1_↩
  digest, void *murmur3_x64_128_digest)

  Finalize the message digests for combined multi-hash and murmur.
- int mh_sha1_murmur3_x64_128_update_base (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer, uint32_t len)

  Combined multi-hash and murmur hash update.
- int mh_sha1_murmur3_x64_128_finalize_base (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_↩
  sha1_digest, void *murmur3_x64_128_digest)

Finalize the message digests for combined multi-hash and murmur.

- int isal_mh_sha1_murmur3_x64_128_init (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const uint64_t murmur_seed)

  Initialize the isal_mh_sha1_murmur3_x64_128_ctx structure.

- int isal_mh_sha1_murmur3_x64_128_update (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer, const uint32_t len)

  Combined multi-hash and murmur hash update.

- int isal_mh_sha1_murmur3_x64_128_finalize (struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_sha1_digest, void *murmur3_x64_128_digest)

  Finalize the message digests for combined multi-hash and murmur.

10.14.1 Detailed Description

mh_sha1_murmur3_x64_128 function prototypes and structures

Interface for mh_sha1_murmur3_x64_128 functions

mh_sha1_murmur3_x64_128 Init-Update..Update-Finalize

This file defines the interface to optimized functions used in mh_sha1 and mh_sha1_murmur3_x64_128. The definition of multi-hash SHA1(mh_sha1, for short) is: Pad the buffer in SHA1 style until the total length is a multiple of 4*16*16(words-width * parallel-segments * block-size); Hash the buffer in parallel, generating digests of 4*16*5 (words-width*parallel-segments* digest-size); Treat the set of digests as another data buffer, and generate a final SHA1 digest for it. mh_sha1_murmur3_x64_128 is a stitching function which will get a murmur3_x64_128 digest while generate mh_sha1 digest.

Example

```
uint32_t mh_sha1_digest[ISAL_SHA1_DIGEST_WORDS];
uint32_t murmur_digest[ISAL_MURMUR3_x64_128_DIGEST_WORDS];
struct isal_mh_sha1_murmur3_x64_128_ctx *ctx;

ctx = malloc(sizeof(struct isal_mh_sha1_murmur3_x64_128_ctx));
isal_mh_sha1_murmur3_x64_128_init(ctx, 0);
isal_mh_sha1_murmur3_x64_128_update(ctx, buff, block_len);
isal_mh_sha1_murmur3_x64_128_finalize(ctx, mh_sha1_digest,
murmur_digest);
```

10.14.2 Enumeration Type Documentation

10.14.2.1 isal_mh_sha1_murmur3_ctx_error

```
enum isal_mh_sha1_murmur3_ctx_error
```

CTX error flags.

Enumerator

ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NONE	ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NONE.
ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NULL	ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NULL.

10.14.3 Function Documentation

10.14.3.1 isal_mh_sha1_murmur3_x64_128_finalize()

```
int isal_mh_sha1_murmur3_x64_128_finalize (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            void * mh_sha1_digest,
            void * murmur3_x64_128_digest )
```

Finalize the message digests for combined multi-hash and murmur.

Place the message digests in mh_sha1_digest and murmur3_x64_128_digest, which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
mh_sha1_digest	The digest of mh_sha1 (5∗4 bytes)
murmur3_x64_128_digest	The digest of murmur3_x64_128 (4∗4 bytes)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.14.3.2 isal_mh_sha1_murmur3_x64_128_init()

```
int isal_mh_sha1_murmur3_x64_128_init (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            const uint64_t murmur_seed )
```

Initialize the isal_mh_sha1_murmur3_x64_128_ctx structure.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
murmur_seed	Seed as an initial digest of murmur3

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.14.3.3 isal_mh_sha1_murmur3_x64_128_update()

```
int isal_mh_sha1_murmur3_x64_128_update (
             struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
             const void * buffer,
             const uint32_t len )
```

Combined multi-hash and murmur hash update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.14.3.4 mh_sha1_murmur3_x64_128_finalize()

```
int mh_sha1_murmur3_x64_128_finalize (
             struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
             void * mh_sha1_digest,
             void * murmur3_x64_128_digest )
```

Finalize the message digests for combined multi-hash and murmur.

Place the message digests in mh_sha1_digest and murmur3_x64_128_digest, which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
mh_sha1_digest	The digest of mh_sha1
murmur3_x64_128_digest	The digest of murmur3_x64_128

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_murmur3_x64_128_finalize() instead.

10.14.3.5 mh_sha1_murmur3_x64_128_finalize_base()

```
int mh_sha1_murmur3_x64_128_finalize_base (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            void * mh_sha1_digest,
            void * murmur3_x64_128_digest )
```

Finalize the message digests for combined multi-hash and murmur.

Place the message digests in mh_sha1_digest and murmur3_x64_128_digest, which must have enough space for the outputs. Base Finalize() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
mh_sha1_digest	The digest of mh_sha1
murmur3_x64_128_digest	The digest of murmur3_x64_128

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_murmur3_x64_128_finalize() instead.

10.14.3.6 mh_sha1_murmur3_x64_128_init()

```
int mh_sha1_murmur3_x64_128_init (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            uint64_t murmur_seed )
```

Initialize the isal_mh_sha1_murmur3_x64_128_ctx structure.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
murmur_seed	Seed as an initial digest of murmur3

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_murmur3_x64_128_init() instead.

10.14.3.7 mh_sha1_murmur3_x64_128_update()

```
int mh_sha1_murmur3_x64_128_update (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Combined multi-hash and murmur hash update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_murmur3_x64_128_update() instead.

10.14.3.8 mh_sha1_murmur3_x64_128_update_base()

```
int mh_sha1_murmur3_x64_128_update_base (
            struct isal_mh_sha1_murmur3_x64_128_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Combined multi-hash and murmur hash update.

Can be called repeatedly to update hashes with new input data. Base update() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha1_murmur3_x64_128 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha1_murmur3_x64_128_update() instead.

10.15 mh_sha1_murmur3_x64_128.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _MH_SHA1_MURMUR3_X64_128_H_
00031 #define _MH_SHA1_MURMUR3_X64_128_H_
00032
00065 #include <stdint.h>
00066 #include "mh_sha1.h"
00067
00068 #ifdef __cplusplus
00069 extern "C" {
00070 #endif
00071
00072 /*
00073  * Define enums from API v2.24, so applications that were using this version
00074  * will still be compiled successfully.
00075  * This list does not need to be extended for new definitions.
00076  */
00077 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00078 /***** Previous hash constants and typedefs *****/
00079 // External Interface Definition
00080 // Add murmur3_x64_128 definition
00081
00082 #define MUR_BLOCK_SIZE               ISAL_MUR_BLOCK_SIZE
00083 #define MURMUR3_x64_128_DIGEST_WORDS ISAL_MURMUR3_x64_128_DIGEST_WORDS
```

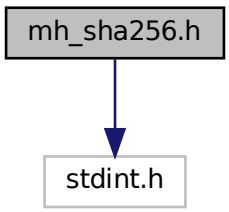
```
00084
00085 #define MH_SHA1_MURMUR3_CTX_ERROR_NONE ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NONE
00086 #define MH_SHA1_MURMUR3_CTX_ERROR_NULL ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NULL
00087
00088 #define mh_sha1_murmur3_x64_128_ctx isal_mh_sha1_murmur3_x64_128_ctx
00089 #define mh_sha1_murmur3_ctx_error   isal_mh_sha1_murmur3_ctx_error
00090 #endif /* NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00091
00092 #define ISAL_MUR_BLOCK_SIZE              (2 * sizeof(uint64_t))
00093 #define ISAL_MURMUR3_x64_128_DIGEST_WORDS 4
00094
00100 struct isal_mh_sha1_murmur3_x64_128_ctx {
00101         uint32_t mh_sha1_digest[ISAL_SHA1_DIGEST_WORDS];
00102         uint32_t murmur3_x64_128_digest[ISAL_MURMUR3_x64_128_DIGEST_WORDS];
00104
00105         uint64_t total_length;
00107         uint8_t partial_block_buffer[ISAL_MH_SHA1_BLOCK_SIZE * 2];
00109         uint8_t mh_sha1_interim_digests[sizeof(uint32_t) * ISAL_SHA1_DIGEST_WORDS * ISAL_HASH_SEGS];
00112         uint8_t frame_buffer[ISAL_MH_SHA1_BLOCK_SIZE + ISAL_AVX512_ALIGNED];
00115 };
00116
00121 enum isal_mh_sha1_murmur3_ctx_error {
00122         ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NONE = 0,
00123         ISAL_MH_SHA1_MURMUR3_CTX_ERROR_NULL = -1,
00124 };
00125
00126 /*****************************************************************
00127  * mh_sha1_murmur3_x64_128 API function prototypes
00128  *****************************************************************/
00129
00138 ISAL_DEPRECATED("Please use isal_mh_sha1_murmur3_x64_128_init() instead")
00139 int
00140 mh_sha1_murmur3_x64_128_init(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, uint64_t murmur_seed);
00141
00155 ISAL_DEPRECATED("Please use isal_mh_sha1_murmur3_x64_128_update() instead")
00156 int
00157 mh_sha1_murmur3_x64_128_update(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, const void *buffer,
00158                                uint32_t len);
00159
00174 ISAL_DEPRECATED("Please use isal_mh_sha1_murmur3_x64_128_finalize() instead")
00175 int
00176 mh_sha1_murmur3_x64_128_finalize(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx, void *mh_sha1_digest,
00177                                  void *murmur3_x64_128_digest);
00178
00179 /*****************************************************************
00180  * multi-types of mh_sha1_murmur3_x64_128 internal API
00181  *
00182  * XXXX         The multi-binary version
00183  * XXXX_base    The C code version which used to display the algorithm
00184  * XXXX_sse     The version uses a ASM function optimized for SSE
00185  * XXXX_avx     The version uses a ASM function optimized for AVX
00186  * XXXX_avx2    The version uses a ASM function optimized for AVX2
00187  *
00188  *****************************************************************/
00189
00202 int
00203 mh_sha1_murmur3_x64_128_update_base(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx,
00204                                     const void *buffer, uint32_t len);
00205
00219 int
00220 mh_sha1_murmur3_x64_128_finalize_base(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx,
00221                                       void *mh_sha1_digest, void *murmur3_x64_128_digest);
00222
00232 int
00233 isal_mh_sha1_murmur3_x64_128_init(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx,
00234                                   const uint64_t murmur_seed);
00235
00250 int
00251 isal_mh_sha1_murmur3_x64_128_update(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx,
00252                                     const void *buffer, const uint32_t len);
00253
00269 int
00270 isal_mh_sha1_murmur3_x64_128_finalize(struct isal_mh_sha1_murmur3_x64_128_ctx *ctx,
00271                                       void *mh_sha1_digest, void *murmur3_x64_128_digest);
00272 #ifdef __cplusplus
00273 }
00274 #endif
00275
00276 #endif
```

10.16 mh_sha256.h File Reference

mh_sha256 function prototypes and structures

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for mh_sha256.h:



Data Structures

- struct isal_mh_sha256_ctx

  Holds info describing a single mh_sha256.

Enumerations

- enum isal_mh_sha256_ctx_error { ISAL_MH_SHA256_CTX_ERROR_NONE = 0 , ISAL_MH_SHA256_CTX_ERROR_NULL = -1 }

  CTX error flags.

Functions

- int mh_sha256_init (struct isal_mh_sha256_ctx *ctx)

  Initialize the isal_mh_sha256_ctx structure.
- int mh_sha256_update (struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha256 update.
- int mh_sha256_finalize (struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest)

  Finalize the message digests for multi-hash sha256.
- int mh_sha256_update_base (struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha256 update.
- int mh_sha256_finalize_base (struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest)

  Finalize the message digests for multi-hash sha256.
- int isal_mh_sha256_init (struct isal_mh_sha256_ctx *ctx)

  Initialize the isal_mh_sha256_ctx structure.
- int isal_mh_sha256_update (struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len)

  Multi-hash sha256 update.
- int isal_mh_sha256_finalize (struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest)

  Finalize the message digests for multi-hash sha256.

10.16.1 Detailed Description

mh_sha256 function prototypes and structures

Interface for mh_sha256 functions

mh_sha256 Init-Update..Update-Finalize

This file defines the interface to optimized functions used in mh_sha256. The definition of multi-hash SHA256(mh_↩ sha256, for short) is: Pad the buffer in SHA256 style until the total length is a multiple of 4∗16∗16 (words-width ∗ parallel-segments ∗ block-size); Hash the buffer in parallel, generating digests of 4∗16∗8 (words-width∗parallel-segments∗ digest-size); Treat the set of digests as another data buffer, and generate a final SHA256 digest for it.

Example

```
uint32_t mh_sha256_digest[ISAL_SHA256_DIGEST_WORDS];
struct isal_mh_sha256_ctx *ctx;

ctx = malloc(sizeof(struct isal_mh_sha256_ctx));
isal_mh_sha256_init(ctx);
isal_mh_sha256_update(ctx, buff, block_len);
isal_mh_sha256_finalize(ctx, mh_sha256_digest);
```

10.16.2 Enumeration Type Documentation

10.16.2.1 isal_mh_sha256_ctx_error

```
enum isal_mh_sha256_ctx_error
```

CTX error flags.

Enumerator

ISAL_MH_SHA256_CTX_ERROR_NONE	ISAL_MH_SHA256_CTX_ERROR_NONE.
ISAL_MH_SHA256_CTX_ERROR_NULL	ISAL_MH_SHA256_CTX_ERROR_NULL.

10.16.3 Function Documentation

10.16.3.1 isal_mh_sha256_finalize()

```
int isal_mh_sha256_finalize (
            struct isal_mh_sha256_ctx * ctx,
            void * mh_sha256_digest )
```

Finalize the message digests for multi-hash sha256.

Place the message digest in mh_sha256_digest which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha256 info
mh_sha256_digest	The digest of mh_sha256

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.16.3.2 isal_mh_sha256_init()

```
int isal_mh_sha256_init (
            struct isal_mh_sha256_ctx * ctx )
```

Initialize the isal_mh_sha256_ctx structure.

Parameters

ctx	Structure holding mh_sha256 info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.16.3.3 isal_mh_sha256_update()

```
int isal_mh_sha256_update (
            struct isal_mh_sha256_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha256 update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha256 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.16.3.4 mh_sha256_finalize()

```
int mh_sha256_finalize (
            struct isal_mh_sha256_ctx * ctx,
            void * mh_sha256_digest )
```

Finalize the message digests for multi-hash sha256.

Place the message digest in mh_sha256_digest which must have enough space for the outputs. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha256 info
mh_sha256_digest	The digest of mh_sha256

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha256_finalize() instead.

10.16.3.5 mh_sha256_finalize_base()

```
int mh_sha256_finalize_base (
            struct isal_mh_sha256_ctx * ctx,
            void * mh_sha256_digest )
```

Finalize the message digests for multi-hash sha256.

Place the message digests in mh_sha256_digest, which must have enough space for the outputs. Base Finalize() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha256 info
mh_sha256_digest	The digest of mh_sha256

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha256_finalize() instead.

10.16.3.6 mh_sha256_init()

```
int mh_sha256_init (
            struct isal_mh_sha256_ctx * ctx )
```

Initialize the isal_mh_sha256_ctx structure.

Parameters

ctx	Structure holding mh_sha256 info

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha256_init() instead.

10.16.3.7 mh_sha256_update()

```
int mh_sha256_update (
            struct isal_mh_sha256_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha256 update.

Can be called repeatedly to update hashes with new input data. This function determines what instruction sets are enabled and selects the appropriate version at runtime.

Parameters

ctx	Structure holding mh_sha256 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha256_update() instead.

10.16.3.8 mh_sha256_update_base()

```
int mh_sha256_update_base (
            struct isal_mh_sha256_ctx * ctx,
            const void * buffer,
            uint32_t len )
```

Multi-hash sha256 update.

Can be called repeatedly to update hashes with new input data. Base update() function that does not require SIMD support.

Parameters

ctx	Structure holding mh_sha256 info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed

Returns

int Return 0 if the function runs without errors

Deprecated Please use isal_mh_sha256_update() instead.

10.17 mh_sha256.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2017 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
```

```
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _MH_SHA256_H_
00031 #define _MH_SHA256_H_
00032
00062 #include <stdint.h>
00063 #include "types.h"
00064
00065 #ifdef __cplusplus
00066 extern "C" {
00067 #endif
00068
00069 /*
00070  * Define enums from API v2.24, so applications that were using this version
00071  * will still be compiled successfully.
00072  * This list does not need to be extended for new definitions.
00073  */
00074 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00075 /***** Previous hash constants and typedefs *****/
00076 #define HASH_SEGS            ISAL_HASH_SEGS
00077 #define SHA256_BLOCK_SIZE    ISAL_SHA256_BLOCK_SIZE
00078 #define MH_SHA256_BLOCK_SIZE ISAL_MH_SHA256_BLOCK_SIZE
00079 #define SHA256_DIGEST_WORDS  ISAL_SHA256_DIGEST_WORDS
00080 #define AVX512_ALIGNED       ISAL_AVX512_ALIGNED
00081
00082 #define MH_SHA256_CTX_ERROR_NONE ISAL_MH_SHA256_CTX_ERROR_NONE
00083 #define MH_SHA256_CTX_ERROR_NULL ISAL_MH_SHA256_CTX_ERROR_NULL
00084
00085 #define mh_sha256_ctx isal_mh_sha256_ctx
00086 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00087
00088 // External Interface Definition
00089 #define ISAL_HASH_SEGS            16
00090 #define ISAL_SHA256_BLOCK_SIZE    64
00091 #define ISAL_MH_SHA256_BLOCK_SIZE (ISAL_HASH_SEGS * ISAL_SHA256_BLOCK_SIZE)
00092 #define ISAL_SHA256_DIGEST_WORDS  8
00093 #define ISAL_AVX512_ALIGNED       64
00094
00100 struct isal_mh_sha256_ctx {
00101         uint32_t mh_sha256_digest[ISAL_SHA256_DIGEST_WORDS];
00102
00103         uint64_t total_length;
00105         uint8_t partial_block_buffer[ISAL_MH_SHA256_BLOCK_SIZE * 2];
00107         uint8_t mh_sha256_interim_digests[sizeof(uint32_t) * ISAL_SHA256_DIGEST_WORDS *
00108                                           ISAL_HASH_SEGS];
00111         uint8_t frame_buffer[ISAL_MH_SHA256_BLOCK_SIZE + ISAL_AVX512_ALIGNED];
00114 };
00115
00120 enum isal_mh_sha256_ctx_error {
00121         ISAL_MH_SHA256_CTX_ERROR_NONE = 0,
00122         ISAL_MH_SHA256_CTX_ERROR_NULL = -1,
00123 };
00124
00125 /*******************************************************************
00126  * mh_sha256 API function prototypes
00127  *******************************************************************/
00128
00136 ISAL_DEPRECATED("Please use isal_mh_sha256_init() instead")
00137 int
00138 mh_sha256_init(struct isal_mh_sha256_ctx *ctx);
00139
00153 ISAL_DEPRECATED("Please use isal_mh_sha256_update() instead")
00154 int
00155 mh_sha256_update(struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len);
00156
00170 ISAL_DEPRECATED("Please use isal_mh_sha256_finalize() instead")
00171 int
00172 mh_sha256_finalize(struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest);
00173
00186 int
00187 mh_sha256_update_base(struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len);
00188
00201 int
00202 mh_sha256_finalize_base(struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest);
```

```
00203
00212 int
00213 isal_mh_sha256_init(struct isal_mh_sha256_ctx *ctx);
00214
00229 int
00230 isal_mh_sha256_update(struct isal_mh_sha256_ctx *ctx, const void *buffer, uint32_t len);
00231
00246 int
00247 isal_mh_sha256_finalize(struct isal_mh_sha256_ctx *ctx, void *mh_sha256_digest);
00248
00249 #ifdef __cplusplus
00250 }
00251 #endif
00252
00253 #endif
```

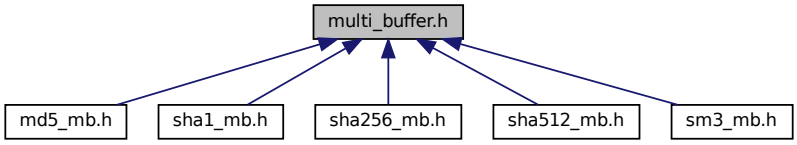
10.18 misc.h

```
00001 /**********************************************************************
00002   Copyright (c) 2021-2023, Intel Corporation
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions are met:
00006
00007       * Redistributions of source code must retain the above copyright notice,
00008         this list of conditions and the following disclaimer.
00009       * Redistributions in binary form must reproduce the above copyright
00010         notice, this list of conditions and the following disclaimer in the
00011         documentation and/or other materials provided with the distribution.
00012       * Neither the name of Intel Corporation nor the names of its contributors
00013         may be used to endorse or promote products derived from this software
00014         without specific prior written permission.
00015
00016   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
00017   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00018   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
00019   DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
00020   FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
00021   DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
00022   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
00023   CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
00024   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00025   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00026 **********************************************************************/
00027
00036 uint64_t
00037 measure_tsc(const uint64_t cycles);
```

10.19 multi_buffer.h File Reference

Multi-buffer common fields.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum ISAL_JOB_STS { ISAL_STS_UNKNOWN = 0 , ISAL_STS_BEING_PROCESSED = 1 , ISAL_STS_COMPLETED = 2 , ISAL_STS_INTERNAL_ERROR , ISAL_STS_ERROR }

  Job return codes.
- enum ISAL_HASH_CTX_FLAG { ISAL_HASH_UPDATE = 0x00 , ISAL_HASH_FIRST = 0x01 , ISAL_HASH_LAST = 0x02 , ISAL_HASH_ENTIRE = 0x03 }

  CTX job type.
- enum ISAL_HASH_CTX_STS { ISAL_HASH_CTX_STS_IDLE = 0x00 , ISAL_HASH_CTX_STS_PROCESSING = 0x01 , ISAL_HASH_CTX_STS_LAST = 0x02 , ISAL_HASH_CTX_STS_COMPLETE = 0x04 }

  CTX status flags.
- enum ISAL_HASH_CTX_ERROR { ISAL_HASH_CTX_ERROR_NONE = 0 , ISAL_HASH_CTX_ERROR_INVALID_FLAGS = -1 , ISAL_HASH_CTX_ERROR_ALREADY_PROCESSING = -2 , ISAL_HASH_CTX_ERROR_ALREADY_COMPLETED = -3 }

  CTX error flags.

10.19.1 Detailed Description

Multi-buffer common fields.

10.19.2 Enumeration Type Documentation

10.19.2.1 ISAL_HASH_CTX_ERROR

```
enum ISAL_HASH_CTX_ERROR
```

CTX error flags.

Enumerator

ISAL_HASH_CTX_ERROR_NONE	ISAL_HASH_CTX_ERROR_NONE.
ISAL_HASH_CTX_ERROR_INVALID_FLAGS	ISAL_HASH_CTX_ERROR_INVALID_FLAGS.
ISAL_HASH_CTX_ERROR_ALREADY_PROCESSING	ISAL_HASH_CTX_ERROR_ALREADY_PROCESSING.
ISAL_HASH_CTX_ERROR_ALREADY_COMPLETED	ISAL_HASH_CTX_ERROR_ALREADY_COMPLETED.

10.19.2.2 ISAL_HASH_CTX_FLAG

```
enum ISAL_HASH_CTX_FLAG
```

CTX job type.

Enumerator

ISAL_HASH_UPDATE	ISAL_HASH_UPDATE.
ISAL_HASH_FIRST	ISAL_HASH_FIRST.
ISAL_HASH_LAST	ISAL_HASH_LAST.
ISAL_HASH_ENTIRE	ISAL_HASH_ENTIRE.

10.19.2.3 ISAL_HASH_CTX_STS

`enum ISAL_HASH_CTX_STS`

CTX status flags.

Enumerator

ISAL_HASH_CTX_STS_IDLE	ISAL_HASH_CTX_STS_IDLE.
ISAL_HASH_CTX_STS_PROCESSING	ISAL_HASH_CTX_STS_PROCESSING.
ISAL_HASH_CTX_STS_LAST	ISAL_HASH_CTX_STS_LAST.
ISAL_HASH_CTX_STS_COMPLETE	ISAL_HASH_CTX_STS_COMPLETE.

10.19.2.4 ISAL_JOB_STS

`enum ISAL_JOB_STS`

Job return codes.

Enumerator

ISAL_STS_UNKNOWN	ISAL_STS_UNKNOWN.
ISAL_STS_BEING_PROCESSED	ISAL_STS_BEING_PROCESSED.
ISAL_STS_COMPLETED	ISAL_STS_COMPLETED.
ISAL_STS_INTERNAL_ERROR	ISAL_STS_INTERNAL_ERROR.
ISAL_STS_ERROR	ISAL_STS_ERROR.

10.20 multi_buffer.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
```

```
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _MULTI_BUFFER_H_
00031 #define _MULTI_BUFFER_H_
00032
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00043 /*
00044  * Define enums from API v2.24, so applications that were using this version
00045  * will still be compiled successfully.
00046  * This list does not need to be extended for new enums.
00047  */
00048 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00049 /***** Previous enums *****/
00050 #define JOB_STS             ISAL_JOB_STS
00051 #define STS_UNKNOWN         ISAL_STS_UNKNOWN
00052 #define STS_BEING_PROCESSED ISAL_STS_BEING_PROCESSED
00053 #define STS_COMPLETED       ISAL_STS_COMPLETED
00054 #define STS_INTERNAL_ERROR  ISAL_STS_INTERNAL_ERROR
00055 #define STS_ERROR           ISAL_STS_ERROR
00056
00057 #define HASH_CTX_FLAG ISAL_HASH_CTX_FLAG
00058 #define HASH_UPDATE   ISAL_HASH_UPDATE
00059 #define HASH_FIRST    ISAL_HASH_FIRST
00060 #define HASH_LAST     ISAL_HASH_LAST
00061 #define HASH_ENTIRE   ISAL_HASH_ENTIRE
00062
00063 #define HASH_CTX_STS            ISAL_HASH_CTX_STS
00064 #define HASH_CTX_STS_IDLE       ISAL_HASH_CTX_STS_IDLE
00065 #define HASH_CTX_STS_PROCESSING ISAL_HASH_CTX_STS_PROCESSING
00066 #define HASH_CTX_STS_LAST       ISAL_HASH_CTX_STS_LAST
00067 #define HASH_CTX_STS_COMPLETE   ISAL_HASH_CTX_STS_COMPLETE
00068
00069 #define HASH_CTX_ERROR                   ISAL_HASH_CTX_ERROR
00070 #define HASH_CTX_ERROR_NONE              ISAL_HASH_CTX_ERROR_NONE
00071 #define HASH_CTX_ERROR_INVALID_FLAGS     ISAL_HASH_CTX_ERROR_INVALID_FLAGS
00072 #define HASH_CTX_ERROR_ALREADY_PROCESSING ISAL_HASH_CTX_ERROR_ALREADY_PROCESSING
00073 #define HASH_CTX_ERROR_ALREADY_COMPLETED  ISAL_HASH_CTX_ERROR_ALREADY_COMPLETED
00074
00075 #define HASH_MB_NO_FLAGS ISAL_HASH_MB_NO_FLAGS
00076 #define HASH_MB_FIRST    ISAL_HASH_MB_FIRST
00077 #define HASH_MB_LAST     ISAL_HASH_MB_LAST
00078
00079 #define hash_ctx_user_data  isal_hash_ctx_user_data
00080 #define hash_ctx_digest     isal_hash_ctx_digest
00081 #define hash_ctx_processing isal_hash_ctx_processing
00082 #define hash_ctx_complete   isal_hash_ctx_complete
00083 #define hash_ctx_status     isal_hash_ctx_status
00084 #define hash_ctx_error      isal_hash_ctx_error
00085 #define hash_ctx_init       isal_hash_ctx_init
00086
00087 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00088
00094 typedef enum {
00095         ISAL_STS_UNKNOWN = 0,
00096         ISAL_STS_BEING_PROCESSED = 1,
00097         ISAL_STS_COMPLETED = 2,
00098         ISAL_STS_INTERNAL_ERROR,
00099         ISAL_STS_ERROR
00100 } ISAL_JOB_STS;
```

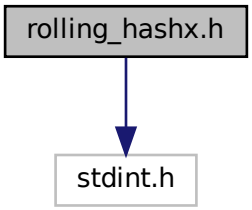
```
00101
00102 #define ISAL_HASH_MB_NO_FLAGS 0
00103 #define ISAL_HASH_MB_FIRST    1
00104 #define ISAL_HASH_MB_LAST     2
00105
00106 /* Common flags for the new API only
00107  *  */
00108
00113 typedef enum {
00114         ISAL_HASH_UPDATE = 0x00,
00115         ISAL_HASH_FIRST = 0x01,
00116         ISAL_HASH_LAST = 0x02,
00117         ISAL_HASH_ENTIRE = 0x03,
00118 } ISAL_HASH_CTX_FLAG;
00119
00124 typedef enum {
00125         ISAL_HASH_CTX_STS_IDLE = 0x00,
00126         ISAL_HASH_CTX_STS_PROCESSING = 0x01,
00127         ISAL_HASH_CTX_STS_LAST = 0x02,
00128         ISAL_HASH_CTX_STS_COMPLETE = 0x04,
00129 } ISAL_HASH_CTX_STS;
00130
00135 typedef enum {
00136         ISAL_HASH_CTX_ERROR_NONE = 0,
00137         ISAL_HASH_CTX_ERROR_INVALID_FLAGS = -1,
00138         ISAL_HASH_CTX_ERROR_ALREADY_PROCESSING = -2,
00139         ISAL_HASH_CTX_ERROR_ALREADY_COMPLETED = -3,
00140 } ISAL_HASH_CTX_ERROR;
00141
00142 #define isal_hash_ctx_user_data(ctx)  ((ctx)->user_data)
00143 #define isal_hash_ctx_digest(ctx)     ((ctx)->job.result_digest)
00144 #define isal_hash_ctx_processing(ctx) ((ctx)->status & ISAL_HASH_CTX_STS_PROCESSING)
00145 #define isal_hash_ctx_complete(ctx)   ((ctx)->status == ISAL_HASH_CTX_STS_COMPLETE)
00146 #define isal_hash_ctx_status(ctx)     ((ctx)->status)
00147 #define isal_hash_ctx_error(ctx)      ((ctx)->error)
00148 #define isal_hash_ctx_init(ctx)                                                    \
00149         do {                                                                       \
00150                 (ctx)->error = ISAL_HASH_CTX_ERROR_NONE;                           \
00151                 (ctx)->status = ISAL_HASH_CTX_STS_COMPLETE;                        \
00152         } while (0)
00153
00154 #ifdef __cplusplus
00155 }
00156 #endif
00157
00158 #endif // _MULTI_BUFFER_H_
```

10.21 rolling_hashx.h File Reference

Fingerprint functions based on rolling hash.

```
#include <stdint.h>
#include "types.h"
```

Include dependency graph for rolling_hashx.h:



Data Structures

- struct isal_rh_state2

  Context for rolling_hash2 functions.

Enumerations

- enum { ISAL_FINGERPRINT_RET_HIT = 0 , ISAL_FINGERPRINT_RET_MAX , ISAL_FINGERPRINT_RET_OTHER }

  rolling hash return values

Functions

- int rolling_hash2_init (struct isal_rh_state2 *state, uint32_t w)

  Initialize state object for rolling hash2.
- void rolling_hash2_reset (struct isal_rh_state2 *state, uint8_t *init_bytes)

  Reset the hash state history.
- int rolling_hash2_run (struct isal_rh_state2 *state, uint8_t *buffer, uint32_t max_len, uint32_t mask, uint32_t trigger, uint32_t *offset)

  Run rolling hash function until trigger met or max length reached.
- uint32_t rolling_hashx_mask_gen (long mean, int shift)

  Generate an appropriate mask to target mean hit rate.
- int isal_rolling_hash2_init (struct isal_rh_state2 *state, const uint32_t w)

  Initialize state object for rolling hash2.
- int isal_rolling_hash2_reset (struct isal_rh_state2 *state, const uint8_t *init_bytes)

  Reset the hash state history.
- int isal_rolling_hash2_run (struct isal_rh_state2 *state, const uint8_t *buffer, const uint32_t max_len, const uint32_t mask, const uint32_t trigger, uint32_t *offset, int *match)

  Run rolling hash function until trigger met or max length reached.
- int isal_rolling_hashx_mask_gen (const uint32_t mean, const uint32_t shift, uint32_t *mask)

  Generate an appropriate mask to target mean hit rate.

10.21.1 Detailed Description

Fingerprint functions based on rolling hash.

rolling_hash2 - checks hash in a sliding window based on random 64-bit hash.

10.21.2 Enumeration Type Documentation

10.21.2.1 anonymous enum

```
anonymous enum
```

rolling hash return values

Enumerator

ISAL_FINGERPRINT_RET_HIT	Fingerprint trigger hit.
ISAL_FINGERPRINT_RET_MAX	Fingerprint max length reached before hit.
ISAL_FINGERPRINT_RET_OTHER	Fingerprint function error returned.

10.21.3 Function Documentation

10.21.3.1 isal_rolling_hash2_init()

```
int isal_rolling_hash2_init (
            struct isal_rh_state2 * state,
            const uint32_t w )
```

Initialize state object for rolling hash2.

Parameters

in	*state*	Structure holding state info on current rolling hash
in	*w*	Window width (1 <= w <= 32)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.21.3.2 isal_rolling_hash2_reset()

```
int isal_rolling_hash2_reset (
            struct isal_rh_state2 * state,
            const uint8_t * init_bytes )
```

Reset the hash state history.

Parameters

in	*state*	Structure holding state info on current rolling hash
in	*init_bytes*	Optional window size buffer to pre-init hash

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.21.3.3 isal_rolling_hash2_run()

```
int isal_rolling_hash2_run (
            struct isal_rh_state2 * state,
            const uint8_t * buffer,
            const uint32_t max_len,
            const uint32_t mask,
            const uint32_t trigger,
            uint32_t * offset,
            int * match )
```

Run rolling hash function until trigger met or max length reached.

Checks for trigger based on a random hash in a sliding window.

Parameters

in	*state*	Structure holding state info on current rolling hash
in	*buffer*	Pointer to input buffer to run windowed hash on
in	*max_len*	Max length to run over input
in	*mask*	Mask bits ORed with hash before test with trigger
in	*trigger*	Match value to compare with windowed hash at each input byte
out	*offset*	Offset from buffer to match, set if match found
out	*match*	Pointer to fingerprint result status to set ISAL_FINGERPRINT_RET_HIT - match found ISAL_FINGERPRINT_RET_MAX - exceeded max length ISAL_FINGERPRINT_RET_OTHER - error

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.21.3.4 isal_rolling_hashx_mask_gen()

```
int isal_rolling_hashx_mask_gen (
            const uint32_t mean,
            const uint32_t shift,
            uint32_t * mask )
```

Generate an appropriate mask to target mean hit rate.

Parameters

in	*mean*	Target chunk size in bytes
in	*shift*	Bits to rotate result to get independent masks
out	*mask*	Generated 32-bit mask value

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.21.3.5 rolling_hash2_init()

```
int rolling_hash2_init (
            struct isal_rh_state2 * state,
            uint32_t w )
```

Initialize state object for rolling hash2.

Parameters

state	Structure holding state info on current rolling hash
w	Window width (1 <= w <= 32)

Returns

0 - success, -1 - failure

Deprecated Please use isal_rolling_hash2_init() instead.

10.21.3.6 rolling_hash2_reset()

```
void rolling_hash2_reset (
            struct isal_rh_state2 * state,
            uint8_t * init_bytes )
```

Reset the hash state history.

Parameters

state	Structure holding state info on current rolling hash
init_bytes	Optional window size buffer to pre-init hash

Returns

none

Deprecated Please use isal_rolling_hash2_reset() instead.

10.21.3.7 rolling_hash2_run()

```
int rolling_hash2_run (
            struct isal_rh_state2 * state,
            uint8_t * buffer,
            uint32_t max_len,
            uint32_t mask,
            uint32_t trigger,
            uint32_t * offset )
```

Run rolling hash function until trigger met or max length reached.

Checks for trigger based on a random hash in a sliding window.

Parameters

state	Structure holding state info on current rolling hash
buffer	Pointer to input buffer to run windowed hash on
max_len	Max length to run over input
mask	Mask bits ORed with hash before test with trigger
trigger	Match value to compare with windowed hash at each input byte
offset	Offset from buffer to match, set if match found

Returns

ISAL_FINGERPRINT_RET_HIT - match found, ISAL_FINGERPRINT_RET_MAX - exceeded max length

Deprecated Please use isal_rolling_hash2_run() instead.

10.21.3.8 rolling_hashx_mask_gen()

```
uint32_t rolling_hashx_mask_gen (
            long mean,
            int shift )
```

Generate an appropriate mask to target mean hit rate.

Parameters

mean	Target chunk size in bytes
shift	Bits to rotate result to get independent masks

Returns

32-bit mask value

Deprecated Please use isal_rolling_hashx_mask_gen() instead.

10.22 rolling_hashx.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2017 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
```

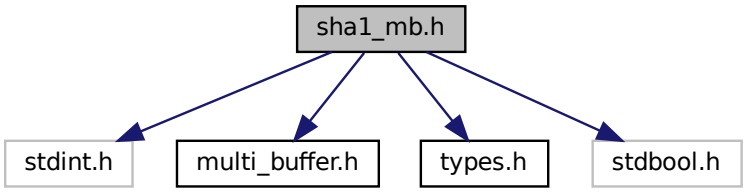
```
00029
00037 #ifndef _ROLLING_HASHX_H_
00038 #define _ROLLING_HASHX_H_
00039
00040 #ifdef __cplusplus
00041 extern "C" {
00042 #endif
00043
00044 #include <stdint.h>
00045 #include "types.h"
00046
00047 /*
00048  * Define enums from API v2.24, so applications that were using this version
00049  * will still be compiled successfully.
00050  * This list does not need to be extended for new definitions.
00051  */
00052 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00053 /***** Previous hash constants and typedefs *****/
00054 #define FINGERPRINT_RET_HIT   ISAL_FINGERPRINT_RET_HIT
00055 #define FINGERPRINT_RET_MAX   ISAL_FINGERPRINT_RET_MAX
00056 #define FINGERPRINT_RET_OTHER ISAL_FINGERPRINT_RET_OTHER
00057
00058 #define FINGERPRINT_MAX_WINDOW ISAL_FINGERPRINT_MAX_WINDOW
00059
00060 #define rh_state2 isal_rh_state2
00061 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00062
00066 enum {
00067         ISAL_FINGERPRINT_RET_HIT = 0,
00068         ISAL_FINGERPRINT_RET_MAX,
00069         ISAL_FINGERPRINT_RET_OTHER
00070 };
00071
00072 #define ISAL_FINGERPRINT_MAX_WINDOW 48
00073
00077 struct isal_rh_state2 {
00078         uint8_t history[ISAL_FINGERPRINT_MAX_WINDOW];
00079         uint64_t table1[256];
00080         uint64_t table2[256];
00081         uint64_t hash;
00082         uint32_t w;
00083 };
00084
00093 ISAL_DEPRECATED("Please use isal_rolling_hash2_init() instead")
00094 int
00095 rolling_hash2_init(struct isal_rh_state2 *state, uint32_t w);
00096
00105 ISAL_DEPRECATED("Please use isal_rolling_hash2_reset() instead")
00106 void
00107 rolling_hash2_reset(struct isal_rh_state2 *state, uint8_t *init_bytes);
00108
00122 ISAL_DEPRECATED("Please use isal_rolling_hash2_run() instead")
00123 int
00124 rolling_hash2_run(struct isal_rh_state2 *state, uint8_t *buffer, uint32_t max_len, uint32_t mask,
00125                   uint32_t trigger, uint32_t *offset);
00126
00135 ISAL_DEPRECATED("Please use isal_rolling_hashx_mask_gen() instead")
00136 uint32_t
00137 rolling_hashx_mask_gen(long mean, int shift);
00138
00148 int
00149 isal_rolling_hash2_init(struct isal_rh_state2 *state, const uint32_t w);
00150
00160 int
00161 isal_rolling_hash2_reset(struct isal_rh_state2 *state, const uint8_t *init_bytes);
00162
00181 int
00182 isal_rolling_hash2_run(struct isal_rh_state2 *state, const uint8_t *buffer, const uint32_t max_len,
00183                        const uint32_t mask, const uint32_t trigger, uint32_t *offset, int *match);
00184
00195 int
00196 isal_rolling_hashx_mask_gen(const uint32_t mean, const uint32_t shift, uint32_t *mask);
00197
00198 #ifdef __cplusplus
00199 }
00200 #endif
00201
00202 #endif // _ROLLING_HASHX_H_
```

10.23 sha1_mb.h File Reference

Multi-buffer CTX API SHA1 function prototypes and structures.

```
#include <stdint.h>
#include <string.h>
#include "multi_buffer.h"
#include "types.h"
#include <stdbool.h>
```

Include dependency graph for sha1_mb.h:



Data Structures

- struct ISAL_SHA1_JOB

  Scheduler layer - Holds info describing a single SHA1 job for the multi-buffer manager.
- struct ISAL_SHA1_MB_ARGS_X16

  Scheduler layer - Holds arguments for submitted SHA1 job.
- struct ISAL_SHA1_LANE_DATA

  Scheduler layer - Lane data.
- struct ISAL_SHA1_MB_JOB_MGR

  Scheduler layer - Holds state for multi-buffer SHA1 jobs.
- struct ISAL_SHA1_HASH_CTX_MGR

  Context layer - Holds state for multi-buffer SHA1 jobs.
- struct ISAL_SHA1_HASH_CTX

  Context layer - Holds info describing a single SHA1 job for the multi-buffer CTX manager. This structure must be allocated to 16-byte aligned memory.

Functions

- void sha1_ctx_mgr_init (ISAL_SHA1_HASH_CTX_MGR *mgr)

  Initialize the SHA1 multi-buffer manager structure.
- ISAL_SHA1_HASH_CTX * sha1_ctx_mgr_submit (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA1 job to the multi-buffer manager.
- ISAL_SHA1_HASH_CTX * sha1_ctx_mgr_flush (ISAL_SHA1_HASH_CTX_MGR *mgr)

Finish all submitted SHA1 jobs and return when complete.

- int isal_sha1_ctx_mgr_init (ISAL_SHA1_HASH_CTX_MGR *mgr)

  Initialize the SHA1 multi-buffer manager structure.

- int isal_sha1_ctx_mgr_submit (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx_in, ISAL_SHA1_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA1 job to the multi-buffer manager.

- int isal_sha1_ctx_mgr_flush (ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX **ctx_out)

  Finish all submitted SHA1 jobs and return when complete.

10.23.1 Detailed Description

Multi-buffer CTX API SHA1 function prototypes and structures.

Warning

: SHA1 is considered unsafe, so it is recommended to use SHA256 instead.

Interface for multi-buffer SHA1 functions

Multi-buffer SHA1 Entire or First-Update..Update-Last

The interface to this multi-buffer hashing code is carried out through the context-level (CTX) init, submit and flush functions and the ISAL_SHA1_HASH_CTX_MGR and ISAL_SHA1_HASH_CTX objects. Numerous ISAL_SHA1_HASH_CTX objects may be instantiated by the application for use with a single ISAL_SHA1_HASH_CTX_MGR.

The CTX interface functions carry out the initialization and padding of the jobs entered by the user and add them to the multi-buffer manager. The lower level "scheduler" layer then processes the jobs in an out-of-order manner. The scheduler layer functions are internal and are not intended to be invoked directly. Jobs can be submitted to a CTX as a complete buffer to be hashed, using the ISAL_HASH_ENTIRE flag, or as partial jobs which can be started using the ISAL_HASH_FIRST flag, and later resumed or finished using the ISAL_HASH_UPDATE and ISAL_HASH_LAST flags respectively.

Note: The submit function does not require data buffers to be block sized.

The SHA1 CTX interface functions are available for 4 architectures: SSE, AVX, AVX2 and AVX512. In addition, a multibinary interface is provided, which selects the appropriate architecture-specific function at runtime.

Usage: The application creates a ISAL_SHA1_HASH_CTX_MGR object and initializes it with a call to sha1_ctx_mgr↩_init*() function, where henceforth "*" stands for the relevant suffix for each architecture; _sse, _avx, _avx2, _avx512(or no suffix for the multibinary version). The ISAL_SHA1_HASH_CTX_MGR object will be used to schedule processor resources, with up to 4 ISAL_SHA1_HASH_CTX objects (or 8 in the AVX2 case, 16 in the AVX512) being processed at a time.

Each ISAL_SHA1_HASH_CTX must be initialized before first use by the isal_hash_ctx_init macro defined in multi_buffer.h. After initialization, the application may begin computing a hash by giving the ISAL_SHA1_HASH_CTX to a ISAL_SHA1_HASH_CTX_MGR using the submit functions sha1_ctx_mgr_submit*() with the ISAL_HASH_FIRST flag set. When the ISAL_SHA1_HASH_CTX is returned to the application (via this or a later call to sha1_ctx_mgr_↩submit*() or sha1_ctx_mgr_flush*()), the application can then re-submit it with another call to sha1_ctx_mgr_submit*(), but without the ISAL_HASH_FIRST flag set.

Ideally, on the last buffer for that hash, sha1_ctx_mgr_submit_sse is called with ISAL_HASH_LAST, although it is also possible to submit the hash with ISAL_HASH_LAST and a zero length if necessary. When a ISAL_SHA1_HASH_CTX is returned after having been submitted with ISAL_HASH_LAST, it will contain a valid hash. The ISAL_SHA1_HASH_CTX can be reused immediately by submitting with ISAL_HASH_FIRST.

For example, you would submit hashes with the following flags for the following numbers of buffers:

- one buffer: ISAL_HASH_FIRST | ISAL_HASH_LAST (or, equivalently, ISAL_HASH_ENTIRE)
- two buffers: ISAL_HASH_FIRST, ISAL_HASH_LAST
- three buffers: ISAL_HASH_FIRST, ISAL_HASH_UPDATE, ISAL_HASH_LAST etc.

The order in which ISAL_SHA1_CTX objects are returned is in general different from the order in which they are submitted.

A few possible error conditions exist:

- Submitting flags other than the allowed entire/first/update/last values
- Submitting a context that is currently being managed by a ISAL_SHA1_HASH_CTX_MGR.
- Submitting a context after ISAL_HASH_LAST is used but before ISAL_HASH_FIRST is set.

These error conditions are reported by returning the ISAL_SHA1_HASH_CTX immediately after a submit with its error member set to a non-zero error code (defined in multi_buffer.h). No changes are made to the ISAL_SHA1_HASH_CTX_MGR in the case of an error; no processing is done for other hashes.

10.23.2 Function Documentation

10.23.2.1 isal_sha1_ctx_mgr_flush()

```
int isal_sha1_ctx_mgr_flush (
            ISAL_SHA1_HASH_CTX_MGR * mgr,
            ISAL_SHA1_HASH_CTX ** ctx_out )
```

Finish all submitted SHA1 jobs and return when complete.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.23.2.2 isal_sha1_ctx_mgr_init()

```
int isal_sha1_ctx_mgr_init (
            ISAL_SHA1_HASH_CTX_MGR * mgr )
```

Initialize the SHA1 multi-buffer manager structure.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.23.2.3 isal_sha1_ctx_mgr_submit()

```
int isal_sha1_ctx_mgr_submit (
            ISAL_SHA1_HASH_CTX_MGR * mgr,
            ISAL_SHA1_HASH_CTX * ctx_in,
            ISAL_SHA1_HASH_CTX ** ctx_out,
            const void * buffer,
            const uint32_t len,
            const ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA1 job to the multi-buffer manager.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
in	*ctx_in*	Structure holding ctx job info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.
in	*buffer*	Pointer to buffer to be processed
in	*len*	Length of buffer (in bytes) to be processed
in	*flags*	Input flag specifying job type (first, update, last or entire)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.23.2.4 sha1_ctx_mgr_flush()

```
ISAL_SHA1_HASH_CTX * sha1_ctx_mgr_flush (
            ISAL_SHA1_HASH_CTX_MGR * mgr )
```

Finish all submitted SHA1 jobs and return when complete.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha1_ctx_mgr_flush() instead.

Parameters

mgr	Structure holding context level state info

Returns

NULL if no jobs to complete or pointer to jobs structure.

10.23.2.5 sha1_ctx_mgr_init()

```
void sha1_ctx_mgr_init (
            ISAL_SHA1_HASH_CTX_MGR * mgr )
```

Initialize the SHA1 multi-buffer manager structure.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha1_ctx_mgr_init() instead.

Parameters

mgr	Structure holding context level state info

Returns

void

10.23.2.6 sha1_ctx_mgr_submit()

```
ISAL_SHA1_HASH_CTX * sha1_ctx_mgr_submit (
            ISAL_SHA1_HASH_CTX_MGR * mgr,
            ISAL_SHA1_HASH_CTX * ctx,
            const void * buffer,
            uint32_t len,
            ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA1 job to the multi-buffer manager.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha1_ctx_mgr_submit() instead.

Parameters

mgr	Structure holding context level state info
ctx	Structure holding ctx job info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed
flags	Input flag specifying job type (first, update, last or entire)

Returns

NULL if no jobs complete or pointer to jobs structure.

10.24 sha1_mb.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
```

```
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _SHA1_MB_H_
00031 #define _SHA1_MB_H_
00032
00108 #include <stdint.h>
00109 #include <string.h>
00110 #include "multi_buffer.h"
00111 #include "types.h"
00112
00113 #ifndef _MSC_VER
00114 #include <stdbool.h>
00115 #endif
00116
00117 #ifdef __cplusplus
00118 extern "C" {
00119 #endif
00120
00121 /*
00122  * Define enums from API v2.24, so applications that were using this version
00123  * will still be compiled successfully.
00124  * This list does not need to be extended for new enums.
00125  */
00126 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00127 /***** Previous hash constants and typedefs *****/
00128 #define SHA1_DIGEST_NWORDS       ISAL_SHA1_DIGEST_NWORDS
00129 #define SHA1_PADLENGTHFIELD_SIZE ISAL_SHA1_PADLENGTHFIELD_SIZE
00130 #define SHA1_MAX_LANES           ISAL_SHA1_MAX_LANES
00131 #define SHA1_MIN_LANES           ISAL_SHA1_MIN_LANES
00132 #define SHA1_BLOCK_SIZE          ISAL_SHA1_BLOCK_SIZE
00133 #define SHA1_WORD_T              ISAL_SHA1_WORD_T
00134
00135 /***** Previous structure definitions *****/
00136 #define SHA1_JOB          ISAL_SHA1_JOB
00137 #define SHA1_MB_ARGS_X16  ISAL_SHA1_MB_ARGS_X16
00138 #define SHA1_LANE_DATA    ISAL_SHA1_LANE_DATA
00139 #define SHA1_MB_JOB_MGR   ISAL_SHA1_MB_JOB_MGR
00140 #define SHA1_HASH_CTX_MGR ISAL_SHA1_HASH_CTX_MGR
00141 #define SHA1_HASH_CTX     ISAL_SHA1_HASH_CTX
00142 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00143
00144 // Hash Constants and Typedefs
00145 #define ISAL_SHA1_DIGEST_NWORDS       5
00146 #define ISAL_SHA1_MAX_LANES           16
00147 #define ISAL_SHA1_MIN_LANES           4
00148 #define ISAL_SHA1_BLOCK_SIZE          64
00149 #define ISAL_SHA1_PADLENGTHFIELD_SIZE 8
00150
00151 typedef uint32_t sha1_digest_array[ISAL_SHA1_DIGEST_NWORDS][ISAL_SHA1_MAX_LANES];
00152 typedef uint32_t ISAL_SHA1_WORD_T;
00153
00156 typedef struct {
00157         uint8_t *buffer;
00158         uint32_t len;
00159         DECLARE_ALIGNED(uint32_t result_digest[ISAL_SHA1_DIGEST_NWORDS], 64);
00160         ISAL_JOB_STS status;
00161         void *user_data;
00162 } ISAL_SHA1_JOB;
00163
00166 typedef struct {
00167         sha1_digest_array digest;
00168         uint8_t *data_ptr[ISAL_SHA1_MAX_LANES];
00169 } ISAL_SHA1_MB_ARGS_X16;
00170
00173 typedef struct {
00174         ISAL_SHA1_JOB *job_in_lane;
00175 } ISAL_SHA1_LANE_DATA;
00176
00179 typedef struct {
00180         ISAL_SHA1_MB_ARGS_X16 args;
```

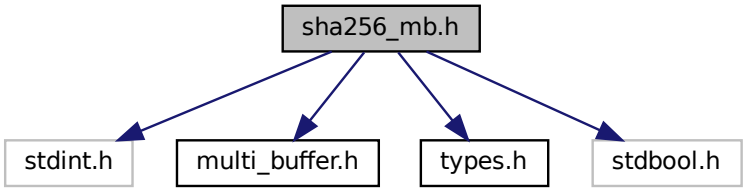
```
00181         DECLARE_ALIGNED(uint32_t lens[ISAL_SHA1_MAX_LANES], 16);
00182         uint64_t unused_lanes;
00184         ISAL_SHA1_LANE_DATA ldata[ISAL_SHA1_MAX_LANES];
00185         uint32_t num_lanes_inuse;
00186 } ISAL_SHA1_MB_JOB_MGR;
00187
00190 typedef struct {
00191         ISAL_SHA1_MB_JOB_MGR mgr;
00192 } ISAL_SHA1_HASH_CTX_MGR;
00193
00197 typedef struct {
00198         ISAL_SHA1_JOB job;              // Must be at struct offset 0.
00199         ISAL_HASH_CTX_STS status;
00200         ISAL_HASH_CTX_ERROR error;
00201         uint64_t total_length;
00202         const void *incoming_buffer;
00203         uint32_t incoming_buffer_length;
00204         uint8_t partial_block_buffer[ISAL_SHA1_BLOCK_SIZE * 2];
00205         uint32_t partial_block_buffer_length;
00206         void *user_data;
00207 } ISAL_SHA1_HASH_CTX;
00208
00209 /******************** multibinary function prototypes **********************/
00210
00219 ISAL_DEPRECATED("Please use isal_sha1_ctx_mgr_init() instead")
00220 void
00221 sha1_ctx_mgr_init(ISAL_SHA1_HASH_CTX_MGR *mgr);
00222
00235 ISAL_DEPRECATED("Please use isal_sha1_ctx_mgr_submit() instead")
00236 ISAL_SHA1_HASH_CTX *
00237 sha1_ctx_mgr_submit(ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx, const void *buffer,
00238                     uint32_t len, ISAL_HASH_CTX_FLAG flags);
00239
00248 ISAL_DEPRECATED("Please use isal_sha1_ctx_mgr_flush() instead")
00249 ISAL_SHA1_HASH_CTX *
00250 sha1_ctx_mgr_flush(ISAL_SHA1_HASH_CTX_MGR *mgr);
00251
00261 int
00262 isal_sha1_ctx_mgr_init(ISAL_SHA1_HASH_CTX_MGR *mgr);
00263
00279 int
00280 isal_sha1_ctx_mgr_submit(ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX *ctx_in,
00281                          ISAL_SHA1_HASH_CTX **ctx_out, const void *buffer, const uint32_t len,
00282                          const ISAL_HASH_CTX_FLAG flags);
00283
00296 int
00297 isal_sha1_ctx_mgr_flush(ISAL_SHA1_HASH_CTX_MGR *mgr, ISAL_SHA1_HASH_CTX **ctx_out);
00298
00299 #ifdef __cplusplus
00300 }
00301 #endif
00302
00303 #endif // _SHA1_MB_H_
```

10.25 sha256_mb.h File Reference

Multi-buffer CTX API SHA256 function prototypes and structures.

```
#include <stdint.h>
#include <string.h>
#include "multi_buffer.h"
#include "types.h"
#include <stdbool.h>
```

Include dependency graph for sha256_mb.h:



Data Structures

- struct ISAL_SHA256_JOB

  Scheduler layer - Holds info describing a single SHA256 job for the multi-buffer manager.
- struct ISAL_SHA256_MB_ARGS_X16

  Scheduler layer - Holds arguments for submitted SHA256 job.
- struct ISAL_SHA256_LANE_DATA

  Scheduler layer - Lane data.
- struct ISAL_SHA256_MB_JOB_MGR

  Scheduler layer - Holds state for multi-buffer SHA256 jobs.
- struct ISAL_SHA256_HASH_CTX_MGR

  Context layer - Holds state for multi-buffer SHA256 jobs.
- struct ISAL_SHA256_HASH_CTX

  Context layer - Holds info describing a single SHA256 job for the multi-buffer CTX manager This structure must be allocated to 16-byte aligned memory.

Functions

- void sha256_ctx_mgr_init (ISAL_SHA256_HASH_CTX_MGR *mgr)

  Initialize the SHA256 multi-buffer manager structure.
- ISAL_SHA256_HASH_CTX * sha256_ctx_mgr_submit (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA256 job to the multi-buffer manager.
- ISAL_SHA256_HASH_CTX * sha256_ctx_mgr_flush (ISAL_SHA256_HASH_CTX_MGR *mgr)

  Finish all submitted SHA256 jobs and return when complete.
- int isal_sha256_ctx_mgr_init (ISAL_SHA256_HASH_CTX_MGR *mgr)

  Initialize the SHA256 multi-buffer manager structure.
- int isal_sha256_ctx_mgr_submit (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx_↩ in, ISAL_SHA256_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA256 job to the multi-buffer manager.
- int isal_sha256_ctx_mgr_flush (ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX **ctx_out)

  Finish all submitted SHA256 jobs and return when complete.

10.25.1 Detailed Description

Multi-buffer CTX API SHA256 function prototypes and structures.

Interface for multi-buffer SHA256 functions

Multi-buffer SHA256 Entire or First-Update..Update-Last

The interface to this multi-buffer hashing code is carried out through the context-level (CTX) init, submit and flush functions and the ISAL_SHA256_HASH_CTX_MGR and ISAL_SHA256_HASH_CTX objects. Numerous ISAL_SHA256_HASH_CTX objects may be instantiated by the application for use with a single ISAL_SHA256_HASH_CTX_MGR.

The CTX interface functions carry out the initialization and padding of the jobs entered by the user and add them to the multi-buffer manager. The lower level "scheduler" layer then processes the jobs in an out-of-order manner. The scheduler layer functions are internal and are not intended to be invoked directly. Jobs can be submitted to a CTX as a complete buffer to be hashed, using the ISAL_HASH_ENTIRE flag, or as partial jobs which can be started using the ISAL_HASH_FIRST flag, and later resumed or finished using the ISAL_HASH_UPDATE and ISAL_HASH_LAST flags respectively.

Note: The submit function does not require data buffers to be block sized.

The SHA256 CTX interface functions are available for 4 architectures: SSE, AVX, AVX2 and AVX512. In addition, a multibinary interface is provided, which selects the appropriate architecture-specific function at runtime.

Usage: The application creates a ISAL_SHA256_HASH_CTX_MGR object and initializes it with a call to sha256_↩ ctx_mgr_init∗() function, where henceforth "∗" stands for the relevant suffix for each architecture; _sse, _avx, _avx2, _avx512(or no suffix for the multibinary version). The ISAL_SHA256_HASH_CTX_MGR object will be used to schedule processor resources, with up to 4 ISAL_SHA256_HASH_CTX objects (or 8 in the AVX2 case, 16 in the AVX512) being processed at a time.

Each ISAL_SHA256_HASH_CTX must be initialized before first use by the isal_hash_ctx_init macro defined in multi_buffer.h. After initialization, the application may begin computing a hash by giving the ISAL_SHA256_HASH_CTX to a ISAL_SHA256_HASH_CTX_MGR using the submit functions sha256_ctx_mgr_submit∗() with the ISAL_HASH_↩ FIRST flag set. When the ISAL_SHA256_HASH_CTX is returned to the application (via this or a later call to sha256↩ _ctx_mgr_submit∗() or sha256_ctx_mgr_flush∗()), the application can then re-submit it with another call to sha256_↩ ctx_mgr_submit∗(), but without the ISAL_HASH_FIRST flag set.

Ideally, on the last buffer for that hash, sha256_ctx_mgr_submit is called with ISAL_HASH_LAST, although it is also possible to submit the hash with ISAL_HASH_LAST and a zero length if necessary. When a ISAL_SHA256_HASH_CTX is returned after having been submitted with ISAL_HASH_LAST, it will contain a valid hash. The ISAL_SHA256_HASH_CTX can be reused immediately by submitting with ISAL_HASH_FIRST.

For example, you would submit hashes with the following flags for the following numbers of buffers:

- one buffer: ISAL_HASH_FIRST | ISAL_HASH_LAST (or, equivalently, ISAL_HASH_ENTIRE)
- two buffers: ISAL_HASH_FIRST, ISAL_HASH_LAST
- three buffers: ISAL_HASH_FIRST, ISAL_HASH_UPDATE, ISAL_HASH_LAST etc.

The order in which SHA256_CTX objects are returned is in general different from the order in which they are submitted.

A few possible error conditions exist:

- Submitting flags other than the allowed entire/first/update/last values
- Submitting a context that is currently being managed by a ISAL_SHA256_HASH_CTX_MGR.
- Submitting a context after ISAL_HASH_LAST is used but before ISAL_HASH_FIRST is set.

These error conditions are reported by returning the ISAL_SHA256_HASH_CTX immediately after a submit with its error member set to a non-zero error code (defined in multi_buffer.h). No changes are made to the ISAL_SHA256_HASH_CTX_MGR in the case of an error; no processing is done for other hashes.

10.25.2 Function Documentation

10.25.2.1 isal_sha256_ctx_mgr_flush()

```
int isal_sha256_ctx_mgr_flush (
            ISAL_SHA256_HASH_CTX_MGR * mgr,
            ISAL_SHA256_HASH_CTX ** ctx_out )
```

Finish all submitted SHA256 jobs and return when complete.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.25.2.2 isal_sha256_ctx_mgr_init()

```
int isal_sha256_ctx_mgr_init (
            ISAL_SHA256_HASH_CTX_MGR * mgr )
```

Initialize the SHA256 multi-buffer manager structure.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.25.2.3 isal_sha256_ctx_mgr_submit()

```
int isal_sha256_ctx_mgr_submit (
            ISAL_SHA256_HASH_CTX_MGR * mgr,
            ISAL_SHA256_HASH_CTX * ctx_in,
            ISAL_SHA256_HASH_CTX ** ctx_out,
            const void * buffer,
            const uint32_t len,
            const ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA256 job to the multi-buffer manager.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
in	*ctx_in*	Structure holding ctx job info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.
in	*buffer*	Pointer to buffer to be processed
in	*len*	Length of buffer (in bytes) to be processed
in	*flags*	Input flag specifying job type (first, update, last or entire)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.25.2.4 sha256_ctx_mgr_flush()

```
ISAL_SHA256_HASH_CTX * sha256_ctx_mgr_flush (
            ISAL_SHA256_HASH_CTX_MGR * mgr )
```

Finish all submitted SHA256 jobs and return when complete.

Requires SSE4.1 or AVX or AVX2

Deprecated Please use isal_sha256_ctx_mgr_flush() instead.

Parameters

mgr	Structure holding context level state info

Returns

NULL if no jobs to complete or pointer to jobs structure.

10.25.2.5 sha256_ctx_mgr_init()

```
void sha256_ctx_mgr_init (
            ISAL_SHA256_HASH_CTX_MGR * mgr )
```

Initialize the SHA256 multi-buffer manager structure.

Requires SSE4.1 or AVX or AVX2

Deprecated Please use isal_sha256_ctx_mgr_init() instead.

Parameters

mgr	Structure holding context level state info

Returns

void

10.25.2.6 sha256_ctx_mgr_submit()

```
ISAL_SHA256_HASH_CTX * sha256_ctx_mgr_submit (
            ISAL_SHA256_HASH_CTX_MGR * mgr,
```

```
        ISAL_SHA256_HASH_CTX * ctx,
        const void * buffer,
        uint32_t len,
        ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA256 job to the multi-buffer manager.

Requires SSE4.1 or AVX or AVX2

Deprecated Please use isal_sha256_ctx_mgr_submit() instead.

Parameters

mgr	Structure holding context level state info
ctx	Structure holding ctx job info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed
flags	Input flag specifying job type (first, update, last or entire)

Returns

NULL if no jobs complete or pointer to jobs structure.

10.26 sha256_mb.h

Go to the documentation of this file.

```
00001 /*******************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 *******************************************************************/
00029
00030 #ifndef _SHA256_MB_H_
00031 #define _SHA256_MB_H_
```

```
00032
00107 #include <stdint.h>
00108 #include <string.h>
00109 #include "multi_buffer.h"
00110 #include "types.h"
00111
00112 #ifndef _MSC_VER
00113 #include <stdbool.h>
00114 #endif
00115
00116 #ifdef __cplusplus
00117 extern "C" {
00118 #endif
00119
00120 /*
00121  * Define enums from API v2.24, so applications that were using this version
00122  * will still be compiled successfully.
00123  * This list does not need to be extended for new definitions.
00124  */
00125 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00126 /***** Previous hash constants and typedefs *****/
00127 #define SHA256_DIGEST_NWORDS       ISAL_SHA256_DIGEST_NWORDS
00128 #define SHA256_PADLENGTHFIELD_SIZE ISAL_SHA256_PADLENGTHFIELD_SIZE
00129 #define SHA256_MAX_LANES           ISAL_SHA256_MAX_LANES
00130 #define SHA256_MIN_LANES           ISAL_SHA256_MIN_LANES
00131 #define SHA256_BLOCK_SIZE          ISAL_SHA256_BLOCK_SIZE
00132 #define SHA256_WORD_T              ISAL_SHA256_WORD_T
00133
00134 /***** Previous structure definitions *****/
00135 #define SHA256_JOB          ISAL_SHA256_JOB
00136 #define SHA256_MB_ARGS_X16  ISAL_SHA256_MB_ARGS_X16
00137 #define SHA256_LANE_DATA    ISAL_SHA256_LANE_DATA
00138 #define SHA256_MB_JOB_MGR   ISAL_SHA256_MB_JOB_MGR
00139 #define SHA256_HASH_CTX_MGR ISAL_SHA256_HASH_CTX_MGR
00140 #define SHA256_HASH_CTX     ISAL_SHA256_HASH_CTX
00141 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00142
00143 // Hash Constants and Typedefs
00144 #define ISAL_SHA256_DIGEST_NWORDS       8
00145 #define ISAL_SHA256_MAX_LANES           16
00146 #define ISAL_SHA256_MIN_LANES           4
00147 #define ISAL_SHA256_BLOCK_SIZE          64
00148 #define ISAL_SHA256_PADLENGTHFIELD_SIZE 8
00149
00150 typedef uint32_t sha256_digest_array[ISAL_SHA256_DIGEST_NWORDS][ISAL_SHA256_MAX_LANES];
00151 typedef uint32_t ISAL_SHA256_WORD_T;
00152
00156 typedef struct {
00157         uint8_t *buffer;
00158         uint64_t len;
00159         DECLARE_ALIGNED(uint32_t result_digest[ISAL_SHA256_DIGEST_NWORDS], 64);
00160         ISAL_JOB_STS status;
00161         void *user_data;
00162 } ISAL_SHA256_JOB;
00163
00166 typedef struct {
00167         sha256_digest_array digest;
00168         uint8_t *data_ptr[ISAL_SHA256_MAX_LANES];
00169 } ISAL_SHA256_MB_ARGS_X16;
00170
00173 typedef struct {
00174         ISAL_SHA256_JOB *job_in_lane;
00175 } ISAL_SHA256_LANE_DATA;
00176
00179 typedef struct {
00180         ISAL_SHA256_MB_ARGS_X16 args;
00181         DECLARE_ALIGNED(uint32_t lens[ISAL_SHA256_MAX_LANES], 16);
00182         uint64_t unused_lanes;
00184         ISAL_SHA256_LANE_DATA ldata[ISAL_SHA256_MAX_LANES];
00185         uint32_t num_lanes_inuse;
00186 } ISAL_SHA256_MB_JOB_MGR;
00187
00190 typedef struct {
00191         ISAL_SHA256_MB_JOB_MGR mgr;
00192 } ISAL_SHA256_HASH_CTX_MGR;
00193
00197 typedef struct {
00198         ISAL_SHA256_JOB job;             // Must be at struct offset 0.
00199         ISAL_HASH_CTX_STS status;
00200         ISAL_HASH_CTX_ERROR error;
00201         uint64_t total_length;
```

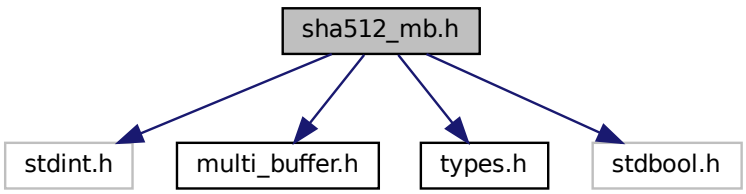
```
00202         const void *incoming_buffer;
00203         uint32_t incoming_buffer_length;
00204         uint8_t partial_block_buffer[ISAL_SHA256_BLOCK_SIZE * 2];
00205         uint32_t partial_block_buffer_length;
00206         void *user_data;
00207 } ISAL_SHA256_HASH_CTX;
00208
00209 /******************** multibinary function prototypes **********************/
00210
00219 ISAL_DEPRECATED("Please use isal_sha256_ctx_mgr_init() instead")
00220 void
00221 sha256_ctx_mgr_init(ISAL_SHA256_HASH_CTX_MGR *mgr);
00222
00235 ISAL_DEPRECATED("Please use isal_sha256_ctx_mgr_submit() instead")
00236 ISAL_SHA256_HASH_CTX *
00237 sha256_ctx_mgr_submit(ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx, const void *buffer,
00238                       uint32_t len, ISAL_HASH_CTX_FLAG flags);
00239
00248 ISAL_DEPRECATED("Please use isal_sha256_ctx_mgr_flush() instead")
00249 ISAL_SHA256_HASH_CTX *
00250 sha256_ctx_mgr_flush(ISAL_SHA256_HASH_CTX_MGR *mgr);
00251
00261 int
00262 isal_sha256_ctx_mgr_init(ISAL_SHA256_HASH_CTX_MGR *mgr);
00263
00280 int
00281 isal_sha256_ctx_mgr_submit(ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX *ctx_in,
00282                            ISAL_SHA256_HASH_CTX **ctx_out, const void *buffer, const uint32_t len,
00283                            const ISAL_HASH_CTX_FLAG flags);
00284
00297 int
00298 isal_sha256_ctx_mgr_flush(ISAL_SHA256_HASH_CTX_MGR *mgr, ISAL_SHA256_HASH_CTX **ctx_out);
00299 #ifdef __cplusplus
00300 }
00301 #endif
00302
00303 #endif // _SHA256_MB_H_
```

10.27 sha512_mb.h File Reference

Single/Multi-buffer CTX API SHA512 function prototypes and structures.

```
#include <stdint.h>
#include <string.h>
#include "multi_buffer.h"
#include "types.h"
#include <stdbool.h>
```

Include dependency graph for sha512_mb.h:



Data Structures

- struct ISAL_SHA512_JOB

  Scheduler layer - Holds info describing a single SHA512 job for the multi-buffer manager.
- struct ISAL_SHA512_MB_ARGS_X8

  Scheduler layer - Holds arguments for submitted SHA512 job.
- struct ISAL_SHA512_LANE_DATA

  Scheduler layer - Lane data.
- struct ISAL_SHA512_MB_JOB_MGR

  Scheduler layer - Holds state for multi-buffer SHA512 jobs.
- struct ISAL_SHA512_HASH_CTX_MGR

  Context layer - Holds state for multi-buffer SHA512 jobs.
- struct ISAL_SHA512_HASH_CTX

  Context layer - Holds info describing a single SHA512 job for the multi-buffer CTX manager.

Functions

- void sha512_ctx_mgr_init (ISAL_SHA512_HASH_CTX_MGR *mgr)

  Initialize the SHA512 multi-buffer manager structure.
- ISAL_SHA512_HASH_CTX * sha512_ctx_mgr_submit (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA512 job to the multi-buffer manager.
- ISAL_SHA512_HASH_CTX * sha512_ctx_mgr_flush (ISAL_SHA512_HASH_CTX_MGR *mgr)

  Finish all submitted SHA512 jobs and return when complete.
- int isal_sha512_ctx_mgr_init (ISAL_SHA512_HASH_CTX_MGR *mgr)

  Initialize the SHA512 multi-buffer manager structure.
- int isal_sha512_ctx_mgr_submit (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx_in, ISAL_SHA512_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

  Submit a new SHA512 job to the multi-buffer manager.
- int isal_sha512_ctx_mgr_flush (ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX **ctx_out)

  Finish all submitted SHA512 jobs and return when complete.

10.27.1 Detailed Description

Single/Multi-buffer CTX API SHA512 function prototypes and structures.

Interface for single and multi-buffer SHA512 functions

Single/Multi-buffer SHA512 Entire or First-Update..Update-Last

The interface to this single/multi-buffer hashing code is carried out through the context-level (CTX) init, submit and flush functions and the ISAL_SHA512_HASH_CTX_MGR and ISAL_SHA512_HASH_CTX objects. Numerous ISAL_SHA512_HASH_CTX objects may be instantiated by the application for use with a single ISAL_SHA512_HASH_CTX_MGR.

The CTX interface functions carry out the initialization and padding of the jobs entered by the user and add them to the multi-buffer manager. The lower level "scheduler" layer then processes the jobs in an out-of-order manner. The

scheduler layer functions are internal and are not intended to be invoked directly. Jobs can be submitted to a CTX as a complete buffer to be hashed, using the ISAL_HASH_ENTIRE flag, or as partial jobs which can be started using the ISAL_HASH_FIRST flag, and later resumed or finished using the ISAL_HASH_UPDATE and ISAL_HASH_LAST flags respectively.

Note: The submit function does not require data buffers to be block sized.

The SHA512 CTX interface functions are available for 5 architectures: multi-buffer SSE, AVX, AVX2, AVX512 and single-buffer SSE4 (which is used in the same way as the multi-buffer code). In addition, a multibinary interface is provided, which selects the appropriate architecture-specific function at runtime. This multibinary interface selects the single buffer SSE4 functions when the platform is detected to be Silvermont.

Usage: The application creates a ISAL_SHA512_HASH_CTX_MGR object and initializes it with a call to sha512_↩ ctx_mgr_init∗() function, where henceforth "∗" stands for the relevant suffix for each architecture; _sse, _avx, _avx2, _avx512(or no suffix for the multibinary version). The ISAL_SHA512_HASH_CTX_MGR object will be used to schedule processor resources, with up to 2 ISAL_SHA512_HASH_CTX objects (or 4 in the AVX2 case, 8 in the AVX512 case) being processed at a time.

Each ISAL_SHA512_HASH_CTX must be initialized before first use by the isal_hash_ctx_init macro defined in multi_buffer.h. After initialization, the application may begin computing a hash by giving the ISAL_SHA512_HASH_CTX to a ISAL_SHA512_HASH_CTX_MGR using the submit functions sha512_ctx_mgr_submit∗() with the ISAL_HASH_↩ FIRST flag set. When the ISAL_SHA512_HASH_CTX is returned to the application (via this or a later call to sha512↩ _ctx_mgr_submit∗() or sha512_ctx_mgr_flush∗()), the application can then re-submit it with another call to sha512_↩ ctx_mgr_submit∗(), but without the ISAL_HASH_FIRST flag set.

Ideally, on the last buffer for that hash, sha512_ctx_mgr_submit is called with ISAL_HASH_LAST, although it is also possible to submit the hash with ISAL_HASH_LAST and a zero length if necessary. When a ISAL_SHA512_HASH_CTX is returned after having been submitted with ISAL_HASH_LAST, it will contain a valid hash. The ISAL_SHA512_HASH_CTX can be reused immediately by submitting with ISAL_HASH_FIRST.

For example, you would submit hashes with the following flags for the following numbers of buffers:

- one buffer: ISAL_HASH_FIRST | ISAL_HASH_LAST (or, equivalently, ISAL_HASH_ENTIRE)
- two buffers: ISAL_HASH_FIRST, ISAL_HASH_LAST
- three buffers: ISAL_HASH_FIRST, ISAL_HASH_UPDATE, ISAL_HASH_LAST etc.

The order in which SHA512_CTX objects are returned is in general different from the order in which they are submitted.

A few possible error conditions exist:

- Submitting flags other than the allowed entire/first/update/last values
- Submitting a context that is currently being managed by a ISAL_SHA512_HASH_CTX_MGR. (Note: This error case is not applicable to the single buffer SSE4 version)
- Submitting a context after ISAL_HASH_LAST is used but before ISAL_HASH_FIRST is set.

These error conditions are reported by returning the ISAL_SHA512_HASH_CTX immediately after a submit with its error member set to a non-zero error code (defined in multi_buffer.h). No changes are made to the ISAL_SHA512_HASH_CTX_MGR in the case of an error; no processing is done for other hashes.

10.27.2 Function Documentation

10.27.2.1 isal_sha512_ctx_mgr_flush()

```
int isal_sha512_ctx_mgr_flush (
            ISAL_SHA512_HASH_CTX_MGR * mgr,
            ISAL_SHA512_HASH_CTX ** ctx_out )
```

Finish all submitted SHA512 jobs and return when complete.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.27.2.2 isal_sha512_ctx_mgr_init()

```
int isal_sha512_ctx_mgr_init (
            ISAL_SHA512_HASH_CTX_MGR * mgr )
```

Initialize the SHA512 multi-buffer manager structure.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.27.2.3 isal_sha512_ctx_mgr_submit()

```
int isal_sha512_ctx_mgr_submit (
            ISAL_SHA512_HASH_CTX_MGR * mgr,
            ISAL_SHA512_HASH_CTX * ctx_in,
            ISAL_SHA512_HASH_CTX ** ctx_out,
            const void * buffer,
            const uint32_t len,
            const ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA512 job to the multi-buffer manager.

Requires SSE4.1 for x86 or ASIMD for ARM

Parameters

in	*mgr*	Structure holding context level state info
in	*ctx_in*	Structure holding ctx job info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.
in	*buffer*	Pointer to buffer to be processed
in	*len*	Length of buffer (in bytes) to be processed
in	*flags*	Input flag specifying job type (first, update, last or entire)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.27.2.4 sha512_ctx_mgr_flush()

```
ISAL_SHA512_HASH_CTX * sha512_ctx_mgr_flush (
            ISAL_SHA512_HASH_CTX_MGR * mgr )
```

Finish all submitted SHA512 jobs and return when complete.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha512_ctx_mgr_flush() instead.

Parameters

mgr	Structure holding context level state info

Returns

NULL if no jobs to complete or pointer to jobs structure.

10.27.2.5 sha512_ctx_mgr_init()

```
void sha512_ctx_mgr_init (
            ISAL_SHA512_HASH_CTX_MGR * mgr )
```

Initialize the SHA512 multi-buffer manager structure.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha512_ctx_mgr_init() instead.

Parameters

mgr	Structure holding context level state info

Returns

void

10.27.2.6 sha512_ctx_mgr_submit()

```
ISAL_SHA512_HASH_CTX * sha512_ctx_mgr_submit (
            ISAL_SHA512_HASH_CTX_MGR * mgr,
```

```
ISAL_SHA512_HASH_CTX * ctx,
const void * buffer,
uint32_t len,
ISAL_HASH_CTX_FLAG flags )
```

Submit a new SHA512 job to the multi-buffer manager.

Requires SSE4.1 or AVX or AVX2 or AVX512

Deprecated Please use isal_sha512_ctx_mgr_submit() instead.

Parameters

mgr	Structure holding context level state info
ctx	Structure holding ctx job info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed
flags	Input flag specifying job type (first, update, last or entire)

Returns

NULL if no jobs complete or pointer to jobs structure.

10.28 sha512_mb.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _SHA512_MB_H_
00031 #define _SHA512_MB_H_
```

```
00032
00110 #include <stdint.h>
00111 #include <string.h>
00112 #include "multi_buffer.h"
00113 #include "types.h"
00114
00115 #ifndef _MSC_VER
00116 #include <stdbool.h>
00117 #endif
00118
00119 #ifdef __cplusplus
00120 extern "C" {
00121 #endif
00122
00123 /*
00124  * Define enums from API v2.24, so applications that were using this version
00125  * will still be compiled successfully.
00126  * This list does not need to be extended for new definitions.
00127  */
00128 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00129 /***** Previous hash constants and typedefs *****/
00130 #define SHA512_DIGEST_NWORDS       ISAL_SHA512_DIGEST_NWORDS
00131 #define SHA512_PADLENGTHFIELD_SIZE ISAL_SHA512_PADLENGTHFIELD_SIZE
00132 #define SHA512_MAX_LANES           ISAL_SHA512_MAX_LANES
00133 #define SHA512_MIN_LANES           ISAL_SHA512_MIN_LANES
00134 #define SHA512_BLOCK_SIZE          ISAL_SHA512_BLOCK_SIZE
00135 #define SHA512_WORD_T              ISAL_SHA512_WORD_T
00136
00137 /***** Previous structure definitions *****/
00138 #define SHA512_JOB          ISAL_SHA512_JOB
00139 #define SHA512_MB_ARGS_X8   ISAL_SHA512_MB_ARGS_X8
00140 #define SHA512_LANE_DATA    ISAL_SHA512_LANE_DATA
00141 #define SHA512_MB_JOB_MGR   ISAL_SHA512_MB_JOB_MGR
00142 #define SHA512_HASH_CTX_MGR ISAL_SHA512_HASH_CTX_MGR
00143 #define SHA512_HASH_CTX     ISAL_SHA512_HASH_CTX
00144 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00145
00146 // Hash Constants and Typedefs
00147 #define ISAL_SHA512_DIGEST_NWORDS       8
00148 #define ISAL_SHA512_MAX_LANES           8
00149 #define ISAL_SHA512_MIN_LANES           2
00150 #define ISAL_SHA512_BLOCK_SIZE          128
00151 #define ISAL_SHA512_PADLENGTHFIELD_SIZE 16
00152
00153 typedef uint64_t sha512_digest_array[ISAL_SHA512_DIGEST_NWORDS][ISAL_SHA512_MAX_LANES];
00154 typedef uint64_t ISAL_SHA512_WORD_T;
00155
00159 typedef struct {
00160         uint8_t *buffer;
00161         uint64_t len;
00162         DECLARE_ALIGNED(uint64_t result_digest[ISAL_SHA512_DIGEST_NWORDS], 64);
00163         ISAL_JOB_STS status;
00164         void *user_data;
00165 } ISAL_SHA512_JOB;
00166
00169 typedef struct {
00170         sha512_digest_array digest;
00171         uint8_t *data_ptr[ISAL_SHA512_MAX_LANES];
00172 } ISAL_SHA512_MB_ARGS_X8;
00173
00176 typedef struct {
00177         ISAL_SHA512_JOB *job_in_lane;
00178 } ISAL_SHA512_LANE_DATA;
00179
00182 typedef struct {
00183         ISAL_SHA512_MB_ARGS_X8 args;
00184         uint64_t lens[ISAL_SHA512_MAX_LANES];
00185         uint64_t unused_lanes;
00187         ISAL_SHA512_LANE_DATA ldata[ISAL_SHA512_MAX_LANES];
00188         uint32_t num_lanes_inuse;
00189 } ISAL_SHA512_MB_JOB_MGR;
00190
00193 typedef struct {
00194         ISAL_SHA512_MB_JOB_MGR mgr;
00195 } ISAL_SHA512_HASH_CTX_MGR;
00196
00200 typedef struct {
00201         ISAL_SHA512_JOB job;             // Must be at struct offset 0.
00202         ISAL_HASH_CTX_STS status;
00203         ISAL_HASH_CTX_ERROR error;
00204         uint64_t total_length;
```

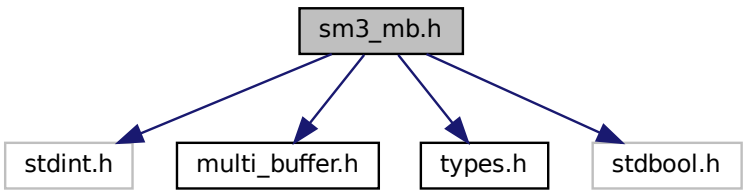
```
00205         const void *incoming_buffer;
00206         uint32_t incoming_buffer_length;
00207         uint8_t partial_block_buffer[ISAL_SHA512_BLOCK_SIZE * 2];
00208         uint32_t partial_block_buffer_length;
00209         void *user_data;
00210 } ISAL_SHA512_HASH_CTX;
00211
00212 /******************** multibinary function prototypes **********************/
00213
00222 ISAL_DEPRECATED("Please use isal_sha512_ctx_mgr_init() instead")
00223 void
00224 sha512_ctx_mgr_init(ISAL_SHA512_HASH_CTX_MGR *mgr);
00225
00238 ISAL_DEPRECATED("Please use isal_sha512_ctx_mgr_submit() instead")
00239 ISAL_SHA512_HASH_CTX *
00240 sha512_ctx_mgr_submit(ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx, const void *buffer,
00241                       uint32_t len, ISAL_HASH_CTX_FLAG flags);
00242
00251 ISAL_DEPRECATED("Please use isal_sha512_ctx_mgr_flush() instead")
00252 ISAL_SHA512_HASH_CTX *
00253 sha512_ctx_mgr_flush(ISAL_SHA512_HASH_CTX_MGR *mgr);
00254
00264 int
00265 isal_sha512_ctx_mgr_init(ISAL_SHA512_HASH_CTX_MGR *mgr);
00266
00283 int
00284 isal_sha512_ctx_mgr_submit(ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX *ctx_in,
00285                            ISAL_SHA512_HASH_CTX **ctx_out, const void *buffer, const uint32_t len,
00286                            const ISAL_HASH_CTX_FLAG flags);
00287
00300 int
00301 isal_sha512_ctx_mgr_flush(ISAL_SHA512_HASH_CTX_MGR *mgr, ISAL_SHA512_HASH_CTX **ctx_out);
00302 #ifdef __cplusplus
00303 }
00304 #endif
00305
00306 #endif // _SHA512_MB_H_
```

10.29 sm3_mb.h File Reference

Multi-buffer CTX API SM3 function prototypes and structures.

```
#include <stdint.h>
#include <string.h>
#include "multi_buffer.h"
#include "types.h"
#include <stdbool.h>
```

Include dependency graph for sm3_mb.h:



Data Structures

- struct ISAL_SM3_JOB

  Scheduler layer - Holds info describing a single SM3 job for the multi-buffer manager.
- struct ISAL_SM3_MB_ARGS_X16

  Scheduler layer - Holds arguments for submitted SM3 job.
- struct ISAL_SM3_LANE_DATA

  Scheduler layer - Lane data.
- struct ISAL_SM3_MB_JOB_MGR

  Scheduler layer - Holds state for multi-buffer SM3 jobs.
- struct ISAL_SM3_HASH_CTX_MGR

  Context layer - Holds state for multi-buffer SM3 jobs.
- struct ISAL_SM3_HASH_CTX

  Context layer - Holds info describing a single SM3 job for the multi-buffer CTX manager.

Functions

- void sm3_ctx_mgr_init (ISAL_SM3_HASH_CTX_MGR *mgr)

  Initialize the SM3 multi-buffer manager structure.
- ISAL_SM3_HASH_CTX * sm3_ctx_mgr_submit (ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX *ctx, const void *buffer, uint32_t len, ISAL_HASH_CTX_FLAG flags)

  Submit a new SM3 job to the multi-buffer manager.
- ISAL_SM3_HASH_CTX * sm3_ctx_mgr_flush (ISAL_SM3_HASH_CTX_MGR *mgr)

  Finish all submitted SM3 jobs and return when complete.
- int isal_sm3_ctx_mgr_init (ISAL_SM3_HASH_CTX_MGR *mgr)

  Initialize the SM3 multi-buffer manager structure.
- int isal_sm3_ctx_mgr_submit (ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX *ctx_in, ISAL_SM3_HASH_CTX **ctx_out, const void *buffer, const uint32_t len, const ISAL_HASH_CTX_FLAG flags)

  Submit a new SM3 job to the multi-buffer manager.
- int isal_sm3_ctx_mgr_flush (ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX **ctx_out)

  Finish all submitted SM3 jobs and return when complete.

10.29.1 Detailed Description

Multi-buffer CTX API SM3 function prototypes and structures.

10.29.2 Function Documentation

10.29.2.1 isal_sm3_ctx_mgr_flush()

```
int isal_sm3_ctx_mgr_flush (
            ISAL_SM3_HASH_CTX_MGR * mgr,
            ISAL_SM3_HASH_CTX ** ctx_out )
```

Finish all submitted SM3 jobs and return when complete.

Parameters

in	*mgr*	Structure holding context level state info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.29.2.2 isal_sm3_ctx_mgr_init()

```
int isal_sm3_ctx_mgr_init (
            ISAL_SM3_HASH_CTX_MGR * mgr )
```

Initialize the SM3 multi-buffer manager structure.

Parameters

mgr	Structure holding context level state info

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.29.2.3 isal_sm3_ctx_mgr_submit()

```
int isal_sm3_ctx_mgr_submit (
            ISAL_SM3_HASH_CTX_MGR * mgr,
            ISAL_SM3_HASH_CTX * ctx_in,
            ISAL_SM3_HASH_CTX ** ctx_out,
            const void * buffer,
```

```
            const uint32_t len,
            const ISAL_HASH_CTX_FLAG flags )
```

Submit a new SM3 job to the multi-buffer manager.

Parameters

in	*mgr*	Structure holding context level state info
in	*ctx_in*	Structure holding ctx job info
out	*ctx_out*	Pointer address to output job ctx info. Modified to point to completed job structure or NULL if no jobs completed.
in	*buffer*	Pointer to buffer to be processed
in	*len*	Length of buffer (in bytes) to be processed
in	*flags*	Input flag specifying job type (first, update, last or entire)

Returns

Operation status

Return values

0	on success
Non-zero	*ISAL_CRYPTO_ERR* on failure

10.29.2.4 sm3_ctx_mgr_flush()

```
ISAL_SM3_HASH_CTX * sm3_ctx_mgr_flush (
            ISAL_SM3_HASH_CTX_MGR * mgr )
```

Finish all submitted SM3 jobs and return when complete.

Parameters

mgr	Structure holding context level state info

Returns

NULL if no jobs to complete or pointer to jobs structure.

Deprecated Please use isal_sm3_ctx_mgr_flush() instead.

10.29.2.5 sm3_ctx_mgr_init()

```
void sm3_ctx_mgr_init (
            ISAL_SM3_HASH_CTX_MGR * mgr )
```

Initialize the SM3 multi-buffer manager structure.

Parameters

mgr	Structure holding context level state info

Returns

void

Deprecated Please use isal_sm3_ctx_mgr_init() instead.

10.29.2.6 sm3_ctx_mgr_submit()

```
ISAL_SM3_HASH_CTX * sm3_ctx_mgr_submit (
            ISAL_SM3_HASH_CTX_MGR * mgr,
            ISAL_SM3_HASH_CTX * ctx,
            const void * buffer,
            uint32_t len,
            ISAL_HASH_CTX_FLAG flags )
```

Submit a new SM3 job to the multi-buffer manager.

Parameters

mgr	Structure holding context level state info
ctx	Structure holding ctx job info
buffer	Pointer to buffer to be processed
len	Length of buffer (in bytes) to be processed
flags	Input flag specifying job type (first, update, last or entire)

Returns

NULL if no jobs complete or pointer to jobs structure.

Deprecated Please use isal_sm3_ctx_mgr_submit() instead.

10.30 sm3_mb.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2020 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
```

```
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00030 #ifndef _SM3_MB_H_
00031 #define _SM3_MB_H_
00032
00038 #include <stdint.h>
00039 #include <string.h>
00040 #include "multi_buffer.h"
00041 #include "types.h"
00042
00043 #ifndef _MSC_VER
00044 #include <stdbool.h>
00045 #endif
00046
00047 #ifdef __cplusplus
00048 extern "C" {
00049 #endif
00050 /*
00051  * Define enums from API v2.24, so applications that were using this version
00052  * will still be compiled successfully.
00053  * This list does not need to be extended for new enums.
00054  */
00055 #ifndef NO_COMPAT_ISAL_CRYPTO_API_2_24
00056 #define SM3_DIGEST_NWORDS ISAL_SM3_DIGEST_NWORDS
00057 #define SM3_MAX_LANES     ISAL_SM3_MAX_LANES
00058 #define SM3_BLOCK_SIZE    ISAL_SM3_BLOCK_SIZE
00059 #define sm3_digest_array  isal_sm3_digest_array
00060 #define SM3_JOB           ISAL_SM3_JOB
00061 #define SM3_MB_ARGS_X16   ISAL_SM3_MB_ARGS_X16
00062 #define SM3_LANE_DATA     ISAL_SM3_LANE_DATA
00063 #define SM3_MB_JOB_MGR    ISAL_SM3_MB_JOB_MGR
00064 #define SM3_HASH_CTX_MGR  ISAL_SM3_HASH_CTX_MGR
00065 #define SM3_HASH_CTX      ISAL_SM3_HASH_CTX
00066 #endif /* !NO_COMPAT_ISAL_CRYPTO_API_2_24 */
00067
00068 #define ISAL_SM3_DIGEST_NWORDS 8 /* Word in SM3 is 32-bit */
00069 #define ISAL_SM3_MAX_LANES     16
00070 #define ISAL_SM3_BLOCK_SIZE    64
00071
00072 typedef uint32_t isal_sm3_digest_array[ISAL_SM3_DIGEST_NWORDS][ISAL_SM3_MAX_LANES];
00073
00076 typedef struct {
00077         uint8_t *buffer;
00078         uint64_t len;
00079         DECLARE_ALIGNED(uint32_t result_digest[ISAL_SM3_DIGEST_NWORDS], 64);
00080         ISAL_JOB_STS status;
00081         void *user_data;
00082 } ISAL_SM3_JOB;
00083
00086 typedef struct {
00087         isal_sm3_digest_array digest;
00088         uint8_t *data_ptr[ISAL_SM3_MAX_LANES];
00089 } ISAL_SM3_MB_ARGS_X16;
00090
00093 typedef struct {
00094         ISAL_SM3_JOB *job_in_lane;
00095 } ISAL_SM3_LANE_DATA;
00096
00099 typedef struct {
00100         ISAL_SM3_MB_ARGS_X16 args;
00101         uint32_t lens[ISAL_SM3_MAX_LANES];
00102         uint64_t unused_lanes;
```

```
00104         ISAL_SM3_LANE_DATA ldata[ISAL_SM3_MAX_LANES];
00105         uint32_t num_lanes_inuse;
00106 } ISAL_SM3_MB_JOB_MGR;
00107
00110 typedef struct {
00111         ISAL_SM3_MB_JOB_MGR mgr;
00112 } ISAL_SM3_HASH_CTX_MGR;
00113
00117 typedef struct {
00118         ISAL_SM3_JOB job;               // Must be at struct offset 0.
00119         ISAL_HASH_CTX_STS status;
00120         ISAL_HASH_CTX_ERROR error;
00121         uint64_t total_length;
00122         const void *incoming_buffer;
00123         uint32_t incoming_buffer_length;
00124         uint8_t partial_block_buffer[ISAL_SM3_BLOCK_SIZE * 2];
00125         uint32_t partial_block_buffer_length;
00126         void *user_data;
00127 } ISAL_SM3_HASH_CTX;
00128
00129 /******************** multibinary function prototypes **********************/
00130
00138 ISAL_DEPRECATED("Please use isal_sm3_ctx_mgr_init() instead.")
00139 void
00140 sm3_ctx_mgr_init(ISAL_SM3_HASH_CTX_MGR *mgr);
00141
00153 ISAL_DEPRECATED("Please use isal_sm3_ctx_mgr_submit() instead.")
00154 ISAL_SM3_HASH_CTX *
00155 sm3_ctx_mgr_submit(ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX *ctx, const void *buffer,
00156                    uint32_t len, ISAL_HASH_CTX_FLAG flags);
00157
00165 ISAL_DEPRECATED("Please use isal_sm3_ctx_mgr_flush() instead.")
00166 ISAL_SM3_HASH_CTX *
00167 sm3_ctx_mgr_flush(ISAL_SM3_HASH_CTX_MGR *mgr);
00168
00177 int
00178 isal_sm3_ctx_mgr_init(ISAL_SM3_HASH_CTX_MGR *mgr);
00179
00195 int
00196 isal_sm3_ctx_mgr_submit(ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX *ctx_in,
00197                         ISAL_SM3_HASH_CTX **ctx_out, const void *buffer, const uint32_t len,
00198                         const ISAL_HASH_CTX_FLAG flags);
00199
00211 int
00212 isal_sm3_ctx_mgr_flush(ISAL_SM3_HASH_CTX_MGR *mgr, ISAL_SM3_HASH_CTX **ctx_out);
00213
00214 #ifdef __cplusplus
00215 }
00216 #endif
00217 #endif
```

10.31 types.h File Reference

Defines common align and debug macros.

This graph shows which files directly or indirectly include this file:

10.31.1 Detailed Description

Defines common align and debug macros.

10.32 types.h

Go to the documentation of this file.

```
00001 /**********************************************************************
00002   Copyright(c) 2011-2016 Intel Corporation All rights reserved.
00003
00004   Redistribution and use in source and binary forms, with or without
00005   modification, are permitted provided that the following conditions
00006   are met:
00007     * Redistributions of source code must retain the above copyright
00008       notice, this list of conditions and the following disclaimer.
00009     * Redistributions in binary form must reproduce the above copyright
00010       notice, this list of conditions and the following disclaimer in
00011       the documentation and/or other materials provided with the
00012       distribution.
00013     * Neither the name of Intel Corporation nor the names of its
00014       contributors may be used to endorse or promote products derived
00015       from this software without specific prior written permission.
00016
00017   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
00018   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
00019   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
00020   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
00021   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00022   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
00023   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
00024   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
00025   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
00026   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
00027   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
00028 **********************************************************************/
00029
00036 #ifndef __TYPES_H
00037 #define __TYPES_H
00038
00039 #ifdef __cplusplus
00040 extern "C" {
00041 #endif
00042
00043 #if defined __unix__
00044 #define DECLARE_ALIGNED(decl, alignval) decl __attribute__((aligned(alignval)))
00045 #define __forceinline                   static inline
00046 #define aligned_free(x)                 free(x)
00047 #else
00048 #ifdef __MINGW32__
00049 #define DECLARE_ALIGNED(decl, alignval) decl __attribute__((aligned(alignval)))
00050 #define posix_memalign(p, algn, len)                                    \
00051         (NULL == (*((char **) (p)) = (void *) _aligned_malloc(len, algn)))
00052 #define aligned_free(x) _aligned_free(x)
00053 #else
00054 #define DECLARE_ALIGNED(decl, alignval) __declspec(align(alignval)) decl
00055 #define posix_memalign(p, algn, len)                                    \
00056         (NULL == (*((char **) (p)) = (void *) _aligned_malloc(len, algn)))
00057 #define aligned_free(x) _aligned_free(x)
00058 #endif
00059 #endif
00060
00061 #ifdef DEBUG
00062 #define DEBUG_PRINT(x) printf x
00063 #else
00064 #define DEBUG_PRINT(x)                                                  \
00065         do {                                                            \
00066         } while (0)
00067 #endif
00068
00069 #ifndef __has_feature
00070 #define __has_feature(x) 0
00071 #endif
00072 #ifndef __has_extension
00073 #define __has_extension __has_feature
00074 #endif
00075 #define ISAL_GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)
00076
00077 #if (defined(__ICC) || defined(__GNUC__) || defined(__clang__)) && !defined(ISAL_UNIT_TEST)
00078 #if __has_extension(attribute_deprecated_with_message) || (ISAL_GCC_VERSION >= 40500) ||          \
00079         (__INTEL_COMPILER >= 1100)
00080 #define ISAL_DEPRECATED(message) __attribute__((deprecated(message)))
00081 #else
00082 #define ISAL_DEPRECATED(message) __attribute__((deprecated))
```

```
00083 #endif
00084 #elif (defined(__ICL) || defined(_MSC_VER))
00085 #if (__INTEL_COMPILER >= 1100) || (_MSC_FULL_VER >= 140050727)
00086 #define ISAL_DEPRECATED(message) __declspec(deprecated(message))
00087 #else
00088 #define ISAL_DEPRECATED(message) __declspec(deprecated)
00089 #endif
00090 #else
00091 #define ISAL_DEPRECATED(message)
00092 #endif
00093
00094 #define ISAL_EXPERIMENTAL(message) ISAL_DEPRECATED("Experimental: " message)
00095
00096 #ifdef __cplusplus
00097 }
00098 #endif
00099
00100 #endif //__TYPES_H
```

10.33 isa-l_crypto.h File Reference

Include for ISA-L_crypto library.

```
#include <isa-l_crypto/aes_cbc.h>
#include <isa-l_crypto/aes_gcm.h>
#include <isa-l_crypto/aes_keyexp.h>
#include <isa-l_crypto/aes_xts.h>
#include <isa-l_crypto/isal_crypto_api.h>
#include <isa-l_crypto/md5_mb.h>
#include <isa-l_crypto/mh_sha1.h>
#include <isa-l_crypto/mh_sha1_murmur3_x64_128.h>
#include <isa-l_crypto/mh_sha256.h>
#include <isa-l_crypto/multi_buffer.h>
#include <isa-l_crypto/rolling_hashx.h>
#include <isa-l_crypto/sha1_mb.h>
#include <isa-l_crypto/sha256_mb.h>
#include <isa-l_crypto/sha512_mb.h>
#include <isa-l_crypto/sm3_mb.h>
```

Include dependency graph for isa-l_crypto.h:

10.33.1 Detailed Description

Include for ISA-L_crypto library.

10.34 isa-l_crypto.h

Go to the documentation of this file.

```
00001
00007 #ifndef _ISAL_CRYPTO_H_
```

```
00008 #define _ISAL_CRYPTO_H_
00009
00010 #include <isa-l_crypto/aes_cbc.h>
00011 #include <isa-l_crypto/aes_gcm.h>
00012 #include <isa-l_crypto/aes_keyexp.h>
00013 #include <isa-l_crypto/aes_xts.h>
00014 #include <isa-l_crypto/isal_crypto_api.h>
00015 #include <isa-l_crypto/md5_mb.h>
00016 #include <isa-l_crypto/mh_sha1.h>
00017 #include <isa-l_crypto/mh_sha1_murmur3_x64_128.h>
00018 #include <isa-l_crypto/mh_sha256.h>
00019 #include <isa-l_crypto/multi_buffer.h>
00020 #include <isa-l_crypto/rolling_hashx.h>
00021 #include <isa-l_crypto/sha1_mb.h>
00022 #include <isa-l_crypto/sha256_mb.h>
00023 #include <isa-l_crypto/sha512_mb.h>
00024 #include <isa-l_crypto/sm3_mb.h>
00025 #endif //_ISAL_CRYPTO_H_
```

Index