

liboqs-cpp

0.1

Generated by Doxygen 1.8.14

Contents

1	liboqs-cpp	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	internal Namespace Reference	11
6.1.1	Detailed Description	11
6.2	oqs Namespace Reference	11
6.2.1	Detailed Description	12
6.2.2	Typedef Documentation	12
6.2.2.1	byte	12
6.2.2.2	bytes	12
6.2.3	Function Documentation	12
6.2.3.1	hex_chop()	12
6.3	oqs::internal Namespace Reference	13
6.4	oqs_literals Namespace Reference	13
6.4.1	Function Documentation	13
6.4.1.1	operator""_bytes()	13

7 Class Documentation	15
7.1 oqs::KeyEncapsulation::alg_details_ Struct Reference	15
7.1.1 Detailed Description	15
7.1.2 Member Data Documentation	15
7.1.2.1 claimed_nist_level	15
7.1.2.2 defined_name	16
7.1.2.3 is_ind_cca	16
7.1.2.4 length_ciphertext	16
7.1.2.5 length_public_key	16
7.1.2.6 length_secret_key	16
7.1.2.7 length_shared_secret	16
7.1.2.8 name	16
7.1.2.9 version	17
7.2 oqs::Signature::alg_details_ Struct Reference	17
7.2.1 Detailed Description	17
7.2.2 Member Data Documentation	17
7.2.2.1 claimed_nist_level	17
7.2.2.2 defined_name	17
7.2.2.3 is_euf_cma	18
7.2.2.4 length_public_key	18
7.2.2.5 length_secret_key	18
7.2.2.6 length_signature	18
7.2.2.7 name	18
7.2.2.8 version	18
7.3 oqs::internal::HexChop Class Reference	18
7.3.1 Detailed Description	19
7.3.2 Constructor & Destructor Documentation	19
7.3.2.1 HexChop()	19
7.3.3 Member Function Documentation	20
7.3.3.1 manipulate_ostream_()	20

7.3.4	Friends And Related Function Documentation	20
7.3.4.1	operator<<	20
7.3.5	Member Data Documentation	20
7.3.5.1	end_	21
7.3.5.2	start_	21
7.3.5.3	v_	21
7.4	oqs::KEMs Class Reference	21
7.4.1	Detailed Description	22
7.4.2	Constructor & Destructor Documentation	23
7.4.2.1	KEMs()	23
7.4.3	Member Function Documentation	23
7.4.3.1	get_enabled_KEMs()	23
7.4.3.2	get_KEM_name()	23
7.4.3.3	get_supported_KEMs()	24
7.4.3.4	is_KEM_enabled()	24
7.4.3.5	is_KEM_supported()	24
7.4.3.6	max_number_KEMs()	25
7.4.4	Friends And Related Function Documentation	25
7.4.4.1	internal::Singleton< const KEMs >	25
7.5	oqs::KeyEncapsulation Class Reference	25
7.5.1	Detailed Description	26
7.5.2	Constructor & Destructor Documentation	26
7.5.2.1	KeyEncapsulation()	26
7.5.2.2	~KeyEncapsulation()	27
7.5.3	Member Function Documentation	27
7.5.3.1	decap_secret()	27
7.5.3.2	encap_secret()	27
7.5.3.3	export_secret_key()	28
7.5.3.4	generate_keypair()	28
7.5.3.5	get_details()	28

7.5.4	Friends And Related Function Documentation	28
7.5.4.1	operator<< [1/2]	28
7.5.4.2	operator<< [2/2]	29
7.5.5	Member Data Documentation	29
7.5.5.1	alg_name_	29
7.5.5.2	details_	29
7.5.5.3	kem_	30
7.5.5.4	secret_key_	30
7.6	oqs::MechanismNotEnabledError Class Reference	30
7.6.1	Detailed Description	31
7.6.2	Constructor & Destructor Documentation	31
7.6.2.1	MechanismNotEnabledError()	31
7.7	oqs::MechanismNotSupportedError Class Reference	32
7.7.1	Detailed Description	32
7.7.2	Constructor & Destructor Documentation	33
7.7.2.1	MechanismNotSupportedError()	33
7.8	oqs::Signature Class Reference	33
7.8.1	Detailed Description	34
7.8.2	Constructor & Destructor Documentation	34
7.8.2.1	Signature()	34
7.8.2.2	~Signature()	35
7.8.3	Member Function Documentation	35
7.8.3.1	export_secret_key()	35
7.8.3.2	generate_keypair()	35
7.8.3.3	get_details()	35
7.8.3.4	sign()	35
7.8.3.5	verify()	36
7.8.4	Friends And Related Function Documentation	36
7.8.4.1	operator<< [1/2]	36
7.8.4.2	operator<< [2/2]	37

7.8.5	Member Data Documentation	37
7.8.5.1	alg_name_	37
7.8.5.2	details_	37
7.8.5.3	secret_key_	37
7.8.5.4	sig_	38
7.9	oqs::Sigs Class Reference	38
7.9.1	Detailed Description	39
7.9.2	Constructor & Destructor Documentation	39
7.9.2.1	Sigs()	39
7.9.3	Member Function Documentation	39
7.9.3.1	get_enabled_sigs()	40
7.9.3.2	get_sig_name()	40
7.9.3.3	get_supported_sigs()	40
7.9.3.4	is_sig_enabled()	40
7.9.3.5	is_sig_supported()	41
7.9.3.6	max_number_sigs()	41
7.9.4	Friends And Related Function Documentation	41
7.9.4.1	internal::Singleton< const Sigs >	41
7.10	oqs::internal::Singleton< T > Class Template Reference	42
7.10.1	Detailed Description	42
7.10.2	Constructor & Destructor Documentation	42
7.10.2.1	Singleton() [1/2]	43
7.10.2.2	Singleton() [2/2]	43
7.10.2.3	~Singleton()	43
7.10.3	Member Function Documentation	43
7.10.3.1	get_instance()	43
7.10.3.2	operator=()	43
8	File Documentation	45
8.1	oqs_cpp.h File Reference	45
8.1.1	Detailed Description	46
8.1.2	Function Documentation	46
8.1.2.1	operator<<() [1/2]	46
8.1.2.2	operator<<() [2/2]	47
	Index	49

Chapter 1

liboqs-cpp

C++ bindings for liboqs

Build status:

liboqs-cpp offers a C++ wrapper for the [Open Quantum Safe liboqs](#) C library. The wrapper is written in standard C++11.

Contents

liboqs-cpp is a header-only wrapper. The project contains the following files and folders:

- `include/oqs_cpp.h`: the main header file
- `examples/kem.cpp`: key encapsulation example
- `examples/sig.cpp`: signature example
- `unit_tests`: unit tests written using Google Test (included)
- `doc`: Doxygen-generated detailed documentation

Usage

To avoid name collisions, liboqs-cpp includes all of its code inside the namespace `oqs`. liboqs-cpp defines four main classes: `oqs::KeyEncapsulation` and `oqs::Signature`, providing post-quantum key encapsulation and signature mechanisms, respectively, and `oqs::KEMs` and `oqs::Sigs`, containing only static member functions that provide information related to the available key encapsulation mechanisms or signature mechanism, respectively.

`oqs::KeyEncapsulation` and/or `oqs::Signature` must be instantiated with a string identifying one of mechanisms supported by liboqs; these can be enumerated using the `oqs::KEMs::get_enabled_KEM_mechanisms()` and `oqs::Sigs::get_enabled_sig_mechanisms()` member functions.

The examples in the `examples` folder provide more details about the wrapper's API.

liboqs installation

liboqs-cpp depends on the [liboqs](#) C library; liboqs must be compiled as a Linux/macOS static library or as a Windows DLL, and be visible to the wrapper, e.g. installed in a system-wide folder.

Compiling on UNIX-like platforms

To use the wrapper, the user must have access to a C++11 compliant compiler, then simply `#include "oqs_+_cpp.h"` in her/his program. The wrapper contains a CMake build system for both examples and unit tests. To compile and run the examples, create a `build` folder inside the root folder of the project, change directory to `build`, then type

```
cmake ..; make -j;
```

The above commands build `oqs_cpp_kem` and `oqs_cpp_sig` examples, assuming the CMake build system is available on the user's platform.

To compile and run the unit tests, first `cd unit_tests`, then create a `build` folder inside `unit_tests`, change directory to it, and finally type

```
cmake ..; make -j;
```

The above commands build `./tests/oqs_cpp_testing` suite of unit tests.

liboqs-cpp has been extensively tested on Linux and macOS systems. Continuous integration is provided via Travis CI.

Compiling on Windows

A Visual Studio solution will be provided soon.

Limitations and security

liboqs is designed for prototyping and evaluating quantum-resistant cryptography. Security of proposed quantum-resistant algorithms may rapidly change as research advances, and may ultimately be completely insecure against either classical or quantum computers.

We believe that the NIST Post-Quantum Cryptography standardization project is currently the best avenue to identifying potentially quantum-resistant algorithms. liboqs does not intend to "pick winners", and we strongly recommend that applications and protocols rely on the outcomes of the NIST standardization project when deploying post-quantum cryptography.

We acknowledge that some parties may want to begin deploying post-quantum cryptography prior to the conclusion of the NIST standardization project. We strongly recommend that any attempts to do make use of so-called **hybrid cryptography**, in which post-quantum public-key algorithms are used alongside traditional public key algorithms (like RSA or elliptic curves) so that the solution is at least no less secure than existing traditional cryptography.

Just like liboqs, liboqs-cpp is provided "as is", without warranty of any kind. See [LICENSE](#) for the full disclaimer.

License

liboqs-cpp is licensed under the MIT License; see [LICENSE](#) for details.

Team

The Open Quantum Safe project is led by [Douglas Stebila](#) and [Michele Mosca](#) at the University of Waterloo.

liboqs-cpp was developed by [Vlad Gheorghiu](#) at evolutionQ and University of Waterloo.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

internal	Internal implementation details	11
oqs	Main namespace for the liboqs C++ wrapper	11
oqs::internal	13
oqs_literals	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

oqs::KeyEncapsulation::alg_details_	15
oqs::Signature::alg_details_	17
oqs::internal::HexChop	18
oqs::KeyEncapsulation	25
runtime_error	
oqs::MechanismNotEnabledError	30
oqs::MechanismNotSupportedError	32
oqs::Signature	33
oqs::internal::Singleton< T >	42
oqs::KEMs	21
oqs::internal::Singleton< const KEMs >	42
oqs::internal::Singleton< const Sigs >	42
oqs::Sigs	38

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

oqs::KeyEncapsulation::alg_details_	
KEM algorithm details	15
oqs::Signature::alg_details_	
Signature algorithm details	17
oqs::internal::HexChop	
Std::ostream manipulator for long vectors of oqs::byte , use it to display only a small number of elements from the beginning and end of the vector	18
oqs::KEMs	
Singleton class, contains details about supported/enabled key exchange mechanisms (KEMs)	21
oqs::KeyEncapsulation	
Key encapsulation mechanisms	25
oqs::MechanismNotEnabledError	
Cryptographic scheme not enabled	30
oqs::MechanismNotSupportedError	
Cryptographic scheme not supported	32
oqs::Signature	
Signature mechanisms	33
oqs::Sigs	
Singleton class, contains details about supported/enabled signature mechanisms	38
oqs::internal::Singleton< T >	
Singleton class using CRTP pattern	42

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

oqs_cpp.h	Main header file for the liboqs C++ wrapper	45
---------------------------	---	--------------------

Chapter 6

Namespace Documentation

6.1 internal Namespace Reference

Internal implementation details.

6.1.1 Detailed Description

Internal implementation details.

6.2 oqs Namespace Reference

Main namespace for the liboqs C++ wrapper.

Namespaces

- [internal](#)

Classes

- class [KEMs](#)
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))
- class [KeyEncapsulation](#)
Key encapsulation mechanisms.
- class [MechanismNotEnabledError](#)
Cryptographic scheme not enabled.
- class [MechanismNotSupportedError](#)
Cryptographic scheme not supported.
- class [Signature](#)
[Signature](#) mechanisms.
- class [Sigs](#)
Singleton class, contains details about supported/enabled signature mechanisms.

Typedefs

- using `byte` = `std::uint8_t`
byte (unsigned)
- using `bytes` = `std::vector< byte >`
vector of bytes (unsigned)

Functions

- `internal::HexChop hex_chop` (`const oqs::bytes &v`, `std::size_t start=8`, `std::size_t end=8`)
Constructs an instance of `oqs::internal::HexChop`.

6.2.1 Detailed Description

Main namespace for the liboqs C++ wrapper.

6.2.2 Typedef Documentation

6.2.2.1 byte

```
using oqs::byte = typedef std::uint8_t
```

byte (unsigned)

6.2.2.2 bytes

```
using oqs::bytes = typedef std::vector<byte>
```

vector of bytes (unsigned)

6.2.3 Function Documentation

6.2.3.1 hex_chop()

```
internal::HexChop oqs::hex_chop (
    const oqs::bytes & v,
    std::size_t start = 8,
    std::size_t end = 8 ) [inline]
```

Constructs an instance of `oqs::internal::HexChop`.

Parameters

<i>v</i>	Vector of bytes
<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector

Returns

Instance of [oqs::internal::HexChop](#)

6.3 oqs::internal Namespace Reference

Classes

- class [HexChop](#)
std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector
- class [Singleton](#)
[Singleton](#) class using CRTP pattern.

6.4 oqs_literals Namespace Reference

Functions

- [oqs::bytes operator""_bytes](#) (const char *c_str, std::size_t length)
User-defined literal operator for converting C-style strings to [oqs::bytes](#).

6.4.1 Function Documentation

6.4.1.1 operator""_bytes()

```
oqs::bytes oqs_literals::operator""_bytes (
    const char * c_str,
    std::size_t length ) [inline]
```

User-defined literal operator for converting C-style strings to [oqs::bytes](#).

Note

The null terminator is not included

Parameters

<i>c_str</i>	C-style string
<i>length</i>	C-style string length (deduced automatically by the compiler)

Returns

The byte representation of the input C-style string

Chapter 7

Class Documentation

7.1 oqs::KeyEncapsulation::alg_details_ Struct Reference

KEM algorithm details.

Public Attributes

- std::string [defined_name](#)
- std::string [name](#)
- std::string [version](#)
- std::size_t [claimed_nist_level](#)
- bool [is_ind_cca](#)
- std::size_t [length_public_key](#)
- std::size_t [length_secret_key](#)
- std::size_t [length_ciphertext](#)
- std::size_t [length_shared_secret](#)

7.1.1 Detailed Description

KEM algorithm details.

7.1.2 Member Data Documentation

7.1.2.1 claimed_nist_level

```
std::size_t oqs::KeyEncapsulation::alg_details_::claimed_nist_level
```

7.1.2.2 defined_name

```
std::string oqs::KeyEncapsulation::alg_details_::defined_name
```

7.1.2.3 is_ind_cca

```
bool oqs::KeyEncapsulation::alg_details_::is_ind_cca
```

7.1.2.4 length_ciphertext

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_ciphertext
```

7.1.2.5 length_public_key

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_public_key
```

7.1.2.6 length_secret_key

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_secret_key
```

7.1.2.7 length_shared_secret

```
std::size_t oqs::KeyEncapsulation::alg_details_::length_shared_secret
```

7.1.2.8 name

```
std::string oqs::KeyEncapsulation::alg_details_::name
```


7.1.2.9 version

```
std::string oqs::KeyEncapsulation::alg_details_::version
```

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

- [oqs_cpp.h](#)

7.2 oqs::Signature::alg_details_ Struct Reference

[Signature](#) algorithm details.

Public Attributes

- std::string [defined_name](#)
- std::string [name](#)
- std::string [version](#)
- std::size_t [claimed_nist_level](#)
- bool [is_euf_cma](#)
- std::size_t [length_public_key](#)
- std::size_t [length_secret_key](#)
- std::size_t [length_signature](#)

7.2.1 Detailed Description

[Signature](#) algorithm details.

7.2.2 Member Data Documentation

7.2.2.1 claimed_nist_level

```
std::size_t oqs::Signature::alg_details_::claimed_nist_level
```

7.2.2.2 defined_name

```
std::string oqs::Signature::alg_details_::defined_name
```

7.2.2.3 is_euf_cma

```
bool oqs::Signature::alg_details_::is_euf_cma
```

7.2.2.4 length_public_key

```
std::size_t oqs::Signature::alg_details_::length_public_key
```

7.2.2.5 length_secret_key

```
std::size_t oqs::Signature::alg_details_::length_secret_key
```

7.2.2.6 length_signature

```
std::size_t oqs::Signature::alg_details_::length_signature
```

7.2.2.7 name

```
std::string oqs::Signature::alg_details_::name
```

7.2.2.8 version

```
std::string oqs::Signature::alg_details_::version
```

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

- [oqs_cpp.h](#)

7.3 oqs::internal::HexChop Class Reference

std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector

```
#include <oqs_cpp.h>
```

Public Member Functions

- [HexChop](#) (const [oqs::bytes](#) &v, std::size_t start, std::size_t end)

Constructs an instance of [oqs::internal::HexChop](#).

Private Member Functions

- void [manipulate_ostream_](#) (std::ostream &os, std::size_t start, std::size_t end, bool is_short) const

std::ostream manipulator

Private Attributes

- [bytes v_](#)

vector of bytes

- std::size_t [start_](#)

- std::size_t [end_](#)

number of hex bytes taken from the start and from the end

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [HexChop](#) &rhs)

std::ostream extraction operator for [oqs::internal::HexChop](#)

7.3.1 Detailed Description

std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector

7.3.2 Constructor & Destructor Documentation

7.3.2.1 HexChop()

```
oqs::internal::HexChop::HexChop (
    const oqs::bytes & v,
    std::size_t start,
    std::size_t end ) [inline], [explicit]
```

Constructs an instance of [oqs::internal::HexChop](#).

Parameters

<i>v</i>	Vector of bytes
<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector

7.3.3 Member Function Documentation

7.3.3.1 `manipulate_ostream_()`

```
void oqs::internal::HexChop::manipulate_ostream_ (
    std::ostream & os,
    std::size_t start,
    std::size_t end,
    bool is_short ) const [inline], [private]
```

`std::ostream` manipulator

Parameters

<i>os</i>	Output stream
<i>start</i>	Number of hex characters displayed from the beginning of the vector
<i>end</i>	Number of hex characters displayed from the end of the vector
<i>is_short</i>	Vector is too short, display all hex characters

7.3.4 Friends And Related Function Documentation

7.3.4.1 `operator<<`

```
std::ostream& operator<< (
    std::ostream & os,
    const HexChop & rhs ) [friend]
```

`std::ostream` extraction operator for `oqs::internal::HexChop`

Parameters

<i>os</i>	Output stream
<i>rhs</i>	<code>oqs::internal::HexChop</code> instance

Returns

Reference to the output stream

7.3.5 Member Data Documentation

7.3.5.1 end_

```
std::size_t oqs::internal::HexChop::end_ [private]
```

number of hex bytes taken from the start and from the end

7.3.5.2 start_

```
std::size_t oqs::internal::HexChop::start_ [private]
```

7.3.5.3 v_

```
bytes oqs::internal::HexChop::v_ [private]
```

vector of bytes

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

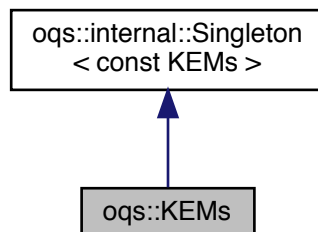
- [oqs_cpp.h](#)

7.4 oqs::KEMs Class Reference

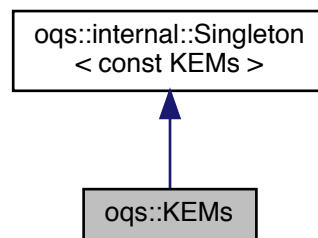
Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::KEMs:



Collaboration diagram for oqs::KEMs:



Static Public Member Functions

- static `std::size_t max_number_KEMs ()`
Maximum number of supported [KEMs](#).
- static `bool is_KEM_supported (const std::string &alg_name)`
Checks whether the KEM algorithm `alg_name` is supported.
- static `bool is_KEM_enabled (const std::string &alg_name)`
Checks whether the KEM algorithm `alg_name` is enabled.
- static `std::string get_KEM_name (std::size_t alg_id)`
KEM algorithm name.
- static `const std::vector< std::string > & get_supported_KEMs ()`
Vector of supported KEM algorithms.
- static `const std::vector< std::string > & get_enabled_KEMs ()`
Vector of enabled KEM algorithms.

Private Member Functions

- `KEMs ()=default`
Private default constructor.

Friends

- class `internal::Singleton< const KEMs >`

Additional Inherited Members

7.4.1 Detailed Description

Singleton class, contains details about supported/enabled key exchange mechanisms ([KEMs](#))

7.4.2 Constructor & Destructor Documentation

7.4.2.1 KEMs()

```
oqs::KEMs::KEMs ( ) [private], [default]
```

Private default constructor.

Note

Use [oqs::KEMs::get_instance\(\)](#) to create an instance

7.4.3 Member Function Documentation

7.4.3.1 get_enabled_KEMs()

```
static const std::vector<std::string>& oqs::KEMs::get_enabled_KEMs ( ) [inline], [static]
```

Vector of enabled KEM algorithms.

Returns

Vector of enabled KEM algorithms

7.4.3.2 get_KEM_name()

```
static std::string oqs::KEMs::get_KEM_name (
    std::size_t alg_id ) [inline], [static]
```

KEM algorithm name.

Parameters

<i>alg_{id}</i>	Cryptographic algorithm numerical id
-------------------------	--------------------------------------

Returns

KEM algorithm name

7.4.3.3 get_supported_KEMs()

```
static const std::vector<std::string>& oqs::KEMs::get_supported_KEMs ( ) [inline], [static]
```

Vector of supported KEM algorithms.

Returns

Vector of supported KEM algorithms

7.4.3.4 is_KEM_enabled()

```
static bool oqs::KEMs::is_KEM_enabled (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the KEM algorithm *alg_name* is enabled.

Parameters

<i>alg_name</i>	Cryptographic algorithm name
-----------------	------------------------------

Returns

True if the KEM algorithm is enabled, false otherwise

7.4.3.5 is_KEM_supported()

```
static bool oqs::KEMs::is_KEM_supported (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the KEM algorithm *alg_name* is supported.

Parameters

<i>alg_name</i>	Cryptographic algorithm name
-----------------	------------------------------

Returns

True if the KEM algorithm is supported, false otherwise

7.4.3.6 max_number_KEMs()

```
static std::size_t oqs::KEMs::max_number_KEMs ( ) [inline], [static]
```

Maximum number of supported [KEMs](#).

Returns

Maximum number of supported [KEMs](#)

7.4.4 Friends And Related Function Documentation

7.4.4.1 internal::Singleton< const KEMs >

```
friend class internal::Singleton< const KEMs > [friend]
```

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

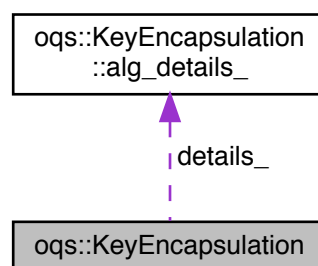
- [oqs_cpp.h](#)

7.5 oqs::KeyEncapsulation Class Reference

Key encapsulation mechanisms.

```
#include <oqs_cpp.h>
```

Collaboration diagram for oqs::KeyEncapsulation:



Classes

- struct [alg_details_](#)
KEM algorithm details.

Public Member Functions

- [KeyEncapsulation](#) (const std::string &alg_name, const [bytes](#) &secret_key={})
Constructs an instance of [oqs::KeyEncapsulation](#).
- virtual [~KeyEncapsulation](#) ()
Virtual default destructor.
- const [alg_details_](#) & [get_details](#) () const
KEM algorithm details.
- [bytes generate_keypair](#) ()
Generate public key/secret key pair.
- [bytes export_secret_key](#) () const
Export secret key.
- std::pair< [bytes](#), [bytes](#) > [encap_secret](#) (const [bytes](#) &public_key) const
Encapsulate secret.
- [bytes decap_secret](#) (const [bytes](#) &ciphertext) const
Decapsulate secret.

Private Attributes

- const std::string [alg_name_](#)
cryptographic algorithm name
- std::shared_ptr< OQS_KEM > [kem_](#)
liboqs smart pointer to OQS_KEM
- [bytes secret_key_](#) {}
secret key
- struct [oqs::KeyEncapsulation::alg_details_ details_](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [alg_details_](#) &rhs)
std::ostream extraction operator for the KEM algorithm details
- std::ostream & [operator<<](#) (std::ostream &os, const [KeyEncapsulation](#) &rhs)
std::ostream extraction operator for [oqs::KeyEncapsulation](#)

7.5.1 Detailed Description

Key encapsulation mechanisms.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 KeyEncapsulation()

```
oqs::KeyEncapsulation::KeyEncapsulation (
    const std::string & alg_name,
    const bytes & secret_key = {} ) [inline], [explicit]
```

Constructs an instance of [oqs::KeyEncapsulation](#).

Parameters

<i>alg_name</i>	Cryptographic algorithm name
<i>secret_key</i>	Secret key (optional)

7.5.2.2 ~KeyEncapsulation()

```
virtual oqs::KeyEncapsulation::~~KeyEncapsulation ( ) [inline], [virtual]
```

Virtual default destructor.

7.5.3 Member Function Documentation

7.5.3.1 decap_secret()

```
bytes oqs::KeyEncapsulation::decap_secret (
    const bytes & ciphertext ) const [inline]
```

Decapsulate secret.

Parameters

<i>ciphertext</i>	Ciphertext
-------------------	------------

Returns

Shared secret

7.5.3.2 encap_secret()

```
std::pair<bytes, bytes> oqs::KeyEncapsulation::encap_secret (
    const bytes & public_key ) const [inline]
```

Encapsulate secret.

Parameters

<i>public_key</i>	Public key
-------------------	------------

Returns

Pair consisting of 1) ciphertext, and 2) shared secret

7.5.3.3 export_secret_key()

```
bytes oqs::KeyEncapsulation::export_secret_key ( ) const [inline]
```

Export secret key.

Returns

Secret key

7.5.3.4 generate_keypair()

```
bytes oqs::KeyEncapsulation::generate_keypair ( ) [inline]
```

Generate public key/secret key pair.

Returns

Public key

7.5.3.5 get_details()

```
const alg_details_& oqs::KeyEncapsulation::get_details ( ) const [inline]
```

KEM algorithm details.

Returns

KEM algorithm details

7.5.4 Friends And Related Function Documentation**7.5.4.1 operator<< [1/2]**

```
std::ostream& operator<< (
    std::ostream & os,
    const alg_details_ & rhs ) [friend]
```

std::ostream extraction operator for the KEM algorithm details

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details instance

Returns

Reference to the output stream

7.5.4.2 operator<< [2/2]

```
std::ostream& operator<< (  
    std::ostream & os,  
    const KeyEncapsulation & rhs ) [friend]
```

std::ostream extraction operator for [oqs::KeyEncapsulation](#)

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Key encapsulation instance

Returns

Reference to the output stream

7.5.5 Member Data Documentation

7.5.5.1 alg_name_

```
const std::string oqs::KeyEncapsulation::alg_name_ [private]
```

cryptographic algorithm name

7.5.5.2 details_

```
struct oqs::KeyEncapsulation::alg_details_ oqs::KeyEncapsulation::details_ [private]
```

7.5.5.3 kem_

```
std::shared_ptr<OQS_KEM> oqs::KeyEncapsulation::kem_ [private]
```

Initial value:

```
{nullptr, [] (OQS_KEM* p) {
                                OQS_KEM_free(p);
}}
```

liboqs smart pointer to OQS_KEM

7.5.5.4 secret_key_

```
bytes oqs::KeyEncapsulation::secret_key_ {} [private]
```

secret key

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

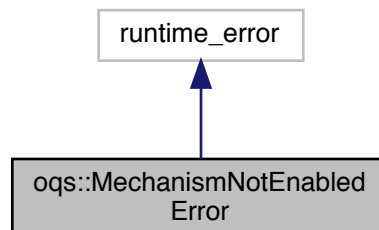
- [oqs_cpp.h](#)

7.6 oqs::MechanismNotEnabledError Class Reference

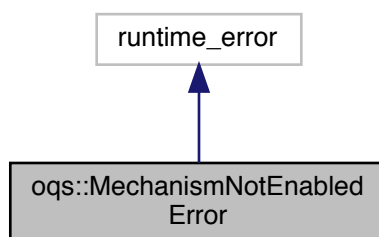
Cryptographic scheme not enabled.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::MechanismNotEnabledError:



Collaboration diagram for oqs::MechanismNotEnabledError:



Public Member Functions

- [MechanismNotEnabledError](#) (const std::string &alg_name)
Constructor.

7.6.1 Detailed Description

Cryptographic scheme not enabled.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 MechanismNotEnabledError()

```
oqs::MechanismNotEnabledError::MechanismNotEnabledError (  
    const std::string & alg_name ) [inline], [explicit]
```

Constructor.

Parameters

<code>alg_name</code>	Cryptographic algorithm name
-----------------------	------------------------------

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

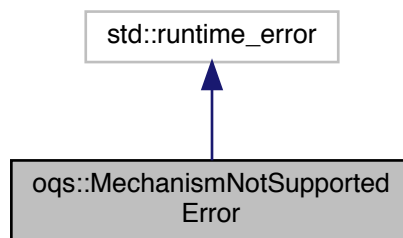
- [oqs_cpp.h](#)

7.7 oqs::MechanismNotSupportedError Class Reference

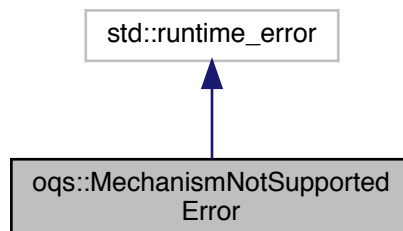
Cryptographic scheme not supported.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::MechanismNotSupportedError:



Collaboration diagram for oqs::MechanismNotSupportedError:



Public Member Functions

- [MechanismNotSupportedError](#) (const std::string &alg_name)
Constructor.

7.7.1 Detailed Description

Cryptographic scheme not supported.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 MechanismNotSupportedError()

```
oqs::MechanismNotSupportedError::MechanismNotSupportedError (
    const std::string & alg_name ) [inline], [explicit]
```

Constructor.

Parameters

<i>alg_name</i>	Cryptographic algorithm name
-----------------	------------------------------

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

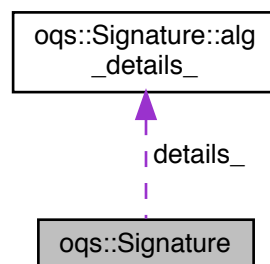
- [oqs_cpp.h](#)

7.8 oqs::Signature Class Reference

[Signature](#) mechanisms.

```
#include <oqs_cpp.h>
```

Collaboration diagram for oqs::Signature:



Classes

- struct [alg_details_](#)
Signature algorithm details.

Public Member Functions

- [Signature](#) (const std::string &alg_name, const [bytes](#) &secret_key={})
Constructs an instance of [oqs::Signature](#).
- virtual [~Signature](#) ()
Virtual default destructor.
- const [alg_details_](#) & [get_details](#) () const
[Signature](#) algorithm details.
- [bytes](#) [generate_keypair](#) ()
Generate public key/secret key pair.
- [bytes](#) [export_secret_key](#) () const
Export secret key.
- [bytes](#) [sign](#) (const [bytes](#) &message)
Sign message.
- bool [verify](#) (const [bytes](#) &message, const [bytes](#) &signature, const [bytes](#) &public_key)
Verify signature.

Private Attributes

- const std::string [alg_name_](#)
cryptographic algorithm name
- std::shared_ptr< OQS_SIG > [sig_](#)
liboqs smart pointer to OQS_SIG
- [bytes](#) [secret_key_](#) {}
secret key
- struct [oqs::Signature::alg_details_](#) [details_](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [alg_details_](#) &rhs)
std::ostream extraction operator for the signature algorithm details
- std::ostream & [operator<<](#) (std::ostream &os, const [Signature](#) &rhs)
std::ostream extraction operator for [oqs::Signature](#)

7.8.1 Detailed Description

[Signature](#) mechanisms.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Signature()

```
oqs::Signature::Signature (
    const std::string & alg_name,
    const bytes & secret_key = {} ) [inline], [explicit]
```

Constructs an instance of [oqs::Signature](#).

Parameters

<i>alg_name</i>	Cryptographic algorithm name
<i>secret_key</i>	Secret key (optional)

7.8.2.2 ~Signature()

```
virtual oqs::Signature::~~Signature ( ) [inline], [virtual]
```

Virtual default destructor.

7.8.3 Member Function Documentation

7.8.3.1 export_secret_key()

```
bytes oqs::Signature::export_secret_key ( ) const [inline]
```

Export secret key.

Returns

Secret key

7.8.3.2 generate_keypair()

```
bytes oqs::Signature::generate_keypair ( ) [inline]
```

Generate public key/secret key pair.

Returns

Public key

7.8.3.3 get_details()

```
const alg_details_& oqs::Signature::get_details ( ) const [inline]
```

[Signature](#) algorithm details.

Returns

[Signature](#) algorithm details

7.8.3.4 sign()

```
bytes oqs::Signature::sign (
    const bytes & message ) [inline]
```

Sign message.

Parameters

<i>message</i>	Message
----------------	---------

Returns

Message signature

7.8.3.5 verify()

```
bool oqs::Signature::verify (
    const bytes & message,
    const bytes & signature,
    const bytes & public_key ) [inline]
```

Verify signature.

Parameters

<i>message</i>	Message
<i>signature</i>	Signature
<i>public_key</i>	Public key

Returns

True if the signature is valid, false otherwise

7.8.4 Friends And Related Function Documentation

7.8.4.1 operator<< [1/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const alg_details_ & rhs ) [friend]
```

std::ostream extraction operator for the signature algorithm details

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Algorithm details

Returns

Reference to the output stream

7.8.4.2 operator<< [2/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const Signature & rhs ) [friend]
```

std::ostream extraction operator for [oqs::Signature](#)

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Signature instance

Returns

Reference to the output stream

7.8.5 Member Data Documentation**7.8.5.1 alg_name_**

```
const std::string oqs::Signature::alg_name_ [private]
```

cryptographic algorithm name

7.8.5.2 details_

```
struct oqs::Signature::alg_details_ oqs::Signature::details_ [private]
```

7.8.5.3 secret_key_

```
bytes oqs::Signature::secret_key_ {} [private]
```

secret key

7.8.5.4 sig_

```
std::shared_ptr<OQS_SIG> oqs::Signature::sig_ [private]
```

Initial value:

```
{nullptr, [] (OQS_SIG* p) {
                                OQS_SIG_free(p);
}}
```

liboqs smart pointer to OQS_SIG

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

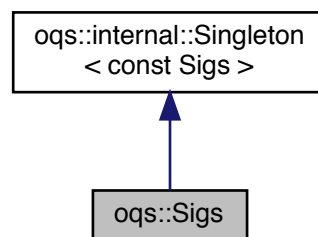
- [oqs_cpp.h](#)

7.9 oqs::Sigs Class Reference

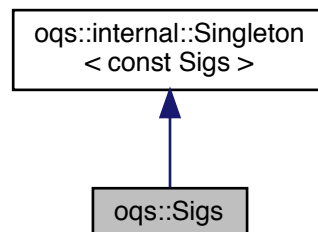
Singleton class, contains details about supported/enabled signature mechanisms.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::Sigs:



Collaboration diagram for oqs::Sigs:



Static Public Member Functions

- static `std::size_t` [max_number_sigs](#) ()
Maximum number of supported signatures.
- static `bool` [is_sig_supported](#) (const `std::string` &alg_name)
Checks whether the signature algorithm alg_name is supported.
- static `bool` [is_sig_enabled](#) (const `std::string` &alg_name)
Checks whether the signature algorithm alg_name is enabled.
- static `std::string` [get_sig_name](#) (`std::size_t` alg_id)
Signature algorithm name.
- static `const std::vector< std::string > &` [get_supported_sigs](#) ()
Vector of supported signature algorithms.
- static `const std::vector< std::string > &` [get_enabled_sigs](#) ()
Vector of enabled signature algorithms.

Private Member Functions

- [Sigs](#) ()=default
Private default constructor.

Friends

- class [internal::Singleton](#)< const [Sigs](#) >

Additional Inherited Members

7.9.1 Detailed Description

Singleton class, contains details about supported/enabled signature mechanisms.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Sigs()

```
oqs::Sigs::Sigs ( ) [private], [default]
```

Private default constructor.

Note

Use [oqs::Sigs::get_instance\(\)](#) to create an instance

7.9.3 Member Function Documentation

7.9.3.1 get_enabled_sigs()

```
static const std::vector<std::string>& oqs::Sigs::get_enabled_sigs ( ) [inline], [static]
```

Vector of enabled signature algorithms.

Returns

Vector of enabled signature algorithms

7.9.3.2 get_sig_name()

```
static std::string oqs::Sigs::get_sig_name (
    std::size_t alg_id ) [inline], [static]
```

[Signature](#) algorithm name.

Parameters

<i>alg</i> ↔ _id	Cryptographic algorithm numerical id
---------------------	--------------------------------------

Returns

[Signature](#) algorithm name

7.9.3.3 get_supported_sigs()

```
static const std::vector<std::string>& oqs::Sigs::get_supported_sigs ( ) [inline], [static]
```

Vector of supported signature algorithms.

Returns

Vector of supported signature algorithms

7.9.3.4 is_sig_enabled()

```
static bool oqs::Sigs::is_sig_enabled (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the signature algorithm *alg_name* is enabled.

Parameters

<i>alg_name</i>	Cryptographic algorithm name
-----------------	------------------------------

Returns

True if the signature algorithm is enabled, false otherwise

7.9.3.5 is_sig_supported()

```
static bool oqs::Sigs::is_sig_supported (
    const std::string & alg_name ) [inline], [static]
```

Checks whether the signature algorithm *alg_name* is supported.

Parameters

<i>alg_name</i>	Cryptographic algorithm name
-----------------	------------------------------

Returns

True if the signature algorithm is supported, false otherwise

7.9.3.6 max_number_sigs()

```
static std::size_t oqs::Sigs::max_number_sigs ( ) [inline], [static]
```

Maximum number of supported signatures.

Returns

Maximum number of supported signatures

7.9.4 Friends And Related Function Documentation

7.9.4.1 internal::Singleton< const Sigs >

```
friend class internal::Singleton< const Sigs > [friend]
```

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

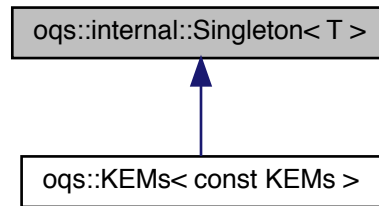
- [oqs_cpp.h](#)

7.10 oqs::internal::Singleton< T > Class Template Reference

[Singleton](#) class using CRTP pattern.

```
#include <oqs_cpp.h>
```

Inheritance diagram for oqs::internal::Singleton< T >:



Static Public Member Functions

- static T & [get_instance](#) () noexcept(std::is_nothrow_constructible< T >::value)
[Singleton](#) instance (thread-safe) via CRTP pattern.

Protected Member Functions

- [Singleton](#) () noexcept=default
- [Singleton](#) (const [Singleton](#) &)=delete
- [Singleton](#) & operator= (const [Singleton](#) &)=delete
- virtual ~[Singleton](#) ()=default

7.10.1 Detailed Description

```
template<typename T>
class oqs::internal::Singleton< T >
```

[Singleton](#) class using CRTP pattern.

Template Parameters

<i>T</i>	Class type of which instance will become a Singleton
----------	--

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Singleton() [1/2]

```
template<typename T>
oqs::internal::Singleton< T >::Singleton ( ) [protected], [default], [noexcept]
```

7.10.2.2 Singleton() [2/2]

```
template<typename T>
oqs::internal::Singleton< T >::Singleton (
    const Singleton< T > & ) [protected], [delete]
```

7.10.2.3 ~Singleton()

```
template<typename T>
virtual oqs::internal::Singleton< T >::~~Singleton ( ) [protected], [virtual], [default]
```

7.10.3 Member Function Documentation

7.10.3.1 get_instance()

```
template<typename T>
static T& oqs::internal::Singleton< T >::get_instance ( ) [inline], [static], [noexcept]
```

Singleton instance (thread-safe) via CRTP pattern.

Note

Code from <https://github.com/vsoftco/qpp/blob/master/include/internal/classes/singleton.h>

Returns

Singleton instance

7.10.3.2 operator=()

```
template<typename T>
Singleton& oqs::internal::Singleton< T >::operator= (
    const Singleton< T > & ) [protected], [delete]
```

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

- [oqs_cpp.h](#)

Chapter 8

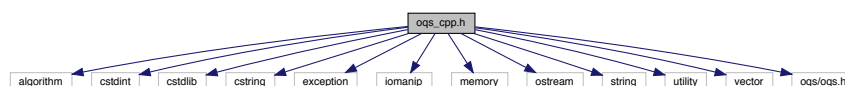
File Documentation

8.1 oqs_cpp.h File Reference

Main header file for the liboqs C++ wrapper.

```
#include <algorithm>
#include <cstdint>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <iomanip>
#include <memory>
#include <ostream>
#include <string>
#include <utility>
#include <vector>
#include <oqs/oqs.h>
```

Include dependency graph for oqs_cpp.h:



Classes

- class [oqs::internal::Singleton< T >](#)
Singleton class using CRTP pattern.
- class [oqs::internal::HexChop](#)
std::ostream manipulator for long vectors of [oqs::byte](#), use it to display only a small number of elements from the beginning and end of the vector
- class [oqs::MechanismNotSupportedError](#)
Cryptographic scheme not supported.
- class [oqs::MechanismNotEnabledError](#)
Cryptographic scheme not enabled.
- class [oqs::KEMs](#)

- Singleton class, contains details about supported/enabled key exchange mechanisms (KEMs)*
 - class [oqs::KeyEncapsulation](#)
Key encapsulation mechanisms.
 - struct [oqs::KeyEncapsulation::alg_details_](#)
KEM algorithm details.
 - class [oqs::Sigs](#)
Singleton class, contains details about supported/enabled signature mechanisms.
 - class [oqs::Signature](#)
Signature mechanisms.
 - struct [oqs::Signature::alg_details_](#)
Signature algorithm details.

Namespaces

- [oqs](#)
Main namespace for the liboqs C++ wrapper.
- [internal](#)
Internal implementation details.
- [oqs::internal](#)
- [oqs_literals](#)

Typedefs

- using [oqs::byte](#) = std::uint8_t
byte (unsigned)
- using [oqs::bytes](#) = std::vector< byte >
vector of bytes (unsigned)

Functions

- internal::HexChop [oqs::hex_chop](#) (const [oqs::bytes](#) &v, std::size_t start=8, std::size_t end=8)
Constructs an instance of [oqs::internal::HexChop](#).
- std::ostream & [operator<<](#) (std::ostream &os, const [oqs::bytes](#) &rhs)
std::ostream extraction operator for [oqs::bytes](#)
- std::ostream & [operator<<](#) (std::ostream &os, const std::vector< std::string > &rhs)
std::ostream extraction operator for vectors of strings
- [oqs::bytes oqs_literals::operator""_bytes](#) (const char *c_str, std::size_t length)
User-defined literal operator for converting C-style strings to [oqs::bytes](#).

8.1.1 Detailed Description

Main header file for the liboqs C++ wrapper.

8.1.2 Function Documentation

8.1.2.1 [operator<<\(\)](#) [1/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const oqs::bytes & rhs ) [inline]
```

std::ostream extraction operator for [oqs::bytes](#)

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Vector of oqs::byte

Returns

Reference to the output stream

8.1.2.2 `operator<<()` [2/2]

```
std::ostream& operator<< (  
    std::ostream & os,  
    const std::vector< std::string > & rhs ) [inline]
```

std::ostream extraction operator for vectors of strings

Parameters

<i>os</i>	Output stream
<i>rhs</i>	Vector of std::string

Returns

Reference to the output stream

Index

- ~KeyEncapsulation
 - oqs::KeyEncapsulation, [27](#)
- ~Signature
 - oqs::Signature, [35](#)
- ~Singleton
 - oqs::internal::Singleton, [43](#)
- alg_name_
 - oqs::KeyEncapsulation, [29](#)
 - oqs::Signature, [37](#)
- byte
 - oqs, [12](#)
- bytes
 - oqs, [12](#)
- claimed_nist_level
 - oqs::KeyEncapsulation::alg_details_, [15](#)
 - oqs::Signature::alg_details_, [17](#)
- decap_secret
 - oqs::KeyEncapsulation, [27](#)
- defined_name
 - oqs::KeyEncapsulation::alg_details_, [15](#)
 - oqs::Signature::alg_details_, [17](#)
- details_
 - oqs::KeyEncapsulation, [29](#)
 - oqs::Signature, [37](#)
- encap_secret
 - oqs::KeyEncapsulation, [27](#)
- end_
 - oqs::internal::HexChop, [20](#)
- export_secret_key
 - oqs::KeyEncapsulation, [28](#)
 - oqs::Signature, [35](#)
- generate_keypair
 - oqs::KeyEncapsulation, [28](#)
 - oqs::Signature, [35](#)
- get_KEM_name
 - oqs::KEMs, [23](#)
- get_details
 - oqs::KeyEncapsulation, [28](#)
 - oqs::Signature, [35](#)
- get_enabled_KEMs
 - oqs::KEMs, [23](#)
- get_enabled_sigs
 - oqs::Sigs, [39](#)
- get_instance
 - oqs::internal::Singleton, [43](#)
- get_sig_name
 - oqs::Sigs, [40](#)
- get_supported_KEMs
 - oqs::KEMs, [23](#)
- get_supported_sigs
 - oqs::Sigs, [40](#)
- hex_chop
 - oqs, [12](#)
- HexChop
 - oqs::internal::HexChop, [19](#)
- internal, [11](#)
- internal::Singleton< const KEMs >
 - oqs::KEMs, [25](#)
- internal::Singleton< const Sigs >
 - oqs::Sigs, [41](#)
- is_KEM_enabled
 - oqs::KEMs, [24](#)
- is_KEM_supported
 - oqs::KEMs, [24](#)
- is_euf_cma
 - oqs::Signature::alg_details_, [17](#)
- is_ind_cca
 - oqs::KeyEncapsulation::alg_details_, [16](#)
- is_sig_enabled
 - oqs::Sigs, [40](#)
- is_sig_supported
 - oqs::Sigs, [41](#)
- KEMs
 - oqs::KEMs, [23](#)
- kem_
 - oqs::KeyEncapsulation, [29](#)
- KeyEncapsulation
 - oqs::KeyEncapsulation, [26](#)
- length_ciphertext
 - oqs::KeyEncapsulation::alg_details_, [16](#)
- length_public_key
 - oqs::KeyEncapsulation::alg_details_, [16](#)
 - oqs::Signature::alg_details_, [18](#)
- length_secret_key
 - oqs::KeyEncapsulation::alg_details_, [16](#)
 - oqs::Signature::alg_details_, [18](#)
- length_shared_secret
 - oqs::KeyEncapsulation::alg_details_, [16](#)
- length_signature
 - oqs::Signature::alg_details_, [18](#)
- manipulate_ostream_

- oqs::internal::HexChop, 20
- max_number_KEMs
 - oqs::KEMs, 24
- max_number_sigs
 - oqs::Sigs, 41
- MechanismNotEnabledError
 - oqs::MechanismNotEnabledError, 31
- MechanismNotSupportedError
 - oqs::MechanismNotSupportedError, 33
- name
 - oqs::KeyEncapsulation::alg_details_, 16
 - oqs::Signature::alg_details_, 18
- operator<<
 - oqs::KeyEncapsulation, 28, 29
 - oqs::Signature, 36, 37
 - oqs::internal::HexChop, 20
 - oqs_cpp.h, 46, 47
- operator=
 - oqs::internal::Singleton, 43
- operator""_bytes
 - oqs_literals, 13
- oqs, 11
 - byte, 12
 - bytes, 12
 - hex_chop, 12
- oqs::KEMs, 21
 - get_KEM_name, 23
 - get_enabled_KEMs, 23
 - get_supported_KEMs, 23
 - internal::Singleton< const KEMs >, 25
 - is_KEM_enabled, 24
 - is_KEM_supported, 24
 - KEMs, 23
 - max_number_KEMs, 24
- oqs::KeyEncapsulation, 25
 - ~KeyEncapsulation, 27
 - alg_name_, 29
 - decap_secret, 27
 - details_, 29
 - encap_secret, 27
 - export_secret_key, 28
 - generate_keypair, 28
 - get_details, 28
 - kem_, 29
 - KeyEncapsulation, 26
 - operator<<, 28, 29
 - secret_key_, 30
- oqs::KeyEncapsulation::alg_details_, 15
 - claimed_nist_level, 15
 - defined_name, 15
 - is_ind_cca, 16
 - length_ciphertext, 16
 - length_public_key, 16
 - length_secret_key, 16
 - length_shared_secret, 16
 - name, 16
 - version, 16
- oqs::MechanismNotEnabledError, 30
 - MechanismNotEnabledError, 31
- oqs::MechanismNotSupportedError, 32
 - MechanismNotSupportedError, 33
- oqs::Signature, 33
 - ~Signature, 35
 - alg_name_, 37
 - details_, 37
 - export_secret_key, 35
 - generate_keypair, 35
 - get_details, 35
 - operator<<, 36, 37
 - secret_key_, 37
 - sig_, 37
 - sign, 35
 - Signature, 34
 - verify, 36
- oqs::Signature::alg_details_, 17
 - claimed_nist_level, 17
 - defined_name, 17
 - is_euf_cma, 17
 - length_public_key, 18
 - length_secret_key, 18
 - length_signature, 18
 - name, 18
 - version, 18
- oqs::Sigs, 38
 - get_enabled_sigs, 39
 - get_sig_name, 40
 - get_supported_sigs, 40
 - internal::Singleton< const Sigs >, 41
 - is_sig_enabled, 40
 - is_sig_supported, 41
 - max_number_sigs, 41
 - Sigs, 39
- oqs::internal, 13
- oqs::internal::HexChop, 18
 - end_, 20
 - HexChop, 19
 - manipulate_ostream_, 20
 - operator<<, 20
 - start_, 21
 - v_, 21
- oqs::internal::Singleton
 - ~Singleton, 43
 - get_instance, 43
 - operator=, 43
 - Singleton, 42, 43
- oqs::internal::Singleton< T >, 42
- oqs_cpp.h, 45
 - operator<<, 46, 47
- oqs_literals, 13
 - operator""_bytes, 13
- secret_key_
 - oqs::KeyEncapsulation, 30
 - oqs::Signature, 37
- sig_
 - oqs::Signature, 37

- sign
 - oqs::Signature, [35](#)
- Signature
 - oqs::Signature, [34](#)
- Sigs
 - oqs::Sigs, [39](#)
- Singleton
 - oqs::internal::Singleton, [42](#), [43](#)
- start_
 - oqs::internal::HexChop, [21](#)
- v_
 - oqs::internal::HexChop, [21](#)
- verify
 - oqs::Signature, [36](#)
- version
 - oqs::KeyEncapsulation::alg_details_, [16](#)
 - oqs::Signature::alg_details_, [18](#)