NAME

SSL_CIPHER_get_name, SSL_CIPHER_standard_name, OPENSSL_cipher_name, SSL_CIPHER_get_bits, SSL_CIPHER_get_version, SSL_CIPHER_description, SSL_CIPHER_get_cipher_nid, SSL_CIPHER_get_digest_nid, SSL_CIPHER_get_handshake_digest, SSL_CIPHER_get_kx_nid, SSL_CIPHER_get_auth_nid, SSL_CIPHER_is_aead, SSL_CIPHER_find, SSL_CIPHER_get_id, SSL_CIPHER_get_protocol_id - get SSL_CIPHER properties

SYNOPSIS

```
#include <openssl/ssl.h>

const char *SSL_CIPHER_get_name(const SSL_CIPHER *cipher);
const char *SSL_CIPHER_standard_name(const SSL_CIPHER *cipher);
const char *OPENSSL_cipher_name(const char *stdname);
int SSL_CIPHER_get_bits(const SSL_CIPHER *cipher, int *alg_bits);
const char *SSL_CIPHER_get_version(const SSL_CIPHER *cipher);
char *SSL_CIPHER_description(const SSL_CIPHER *cipher, char *buf, int size);
int SSL_CIPHER_get_cipher_nid(const SSL_CIPHER *c);
int SSL_CIPHER_get_digest_nid(const SSL_CIPHER *c);
const EVP_MD *SSL_CIPHER_get_handshake_digest(const SSL_CIPHER *c);
int SSL_CIPHER_get_kx_nid(const SSL_CIPHER *c);
int SSL_CIPHER_get_auth_nid(const SSL_CIPHER *c);
int SSL_CIPHER_is_aead(const SSL_CIPHER *c);
const SSL_CIPHER *SSL_CIPHER_find(SSL *ssl, const unsigned char *ptr);
uint32_t SSL_CIPHER_get_id(const SSL_CIPHER *c);
uint32_t SSL_CIPHER_get_protocol_id(const SSL_CIPHER *c);
```

DESCRIPTION

SSL_CIPHER_get_name() returns a pointer to the name of **cipher**. If the **cipher** is NULL, it returns "(NONE)".

SSL_CIPHER_standard_name() returns a pointer to the standard RFC name of **cipher**. If the **cipher** is NULL, it returns "(NONE)". If the **cipher** has no standard name, it returns **NULL**. If **cipher** was defined in both SSLv3 and TLS, it returns the TLS name.

OPENSSL_cipher_name() returns a pointer to the OpenSSL name of **stdname**. If the **stdname** is NULL, or **stdname** has no corresponding OpenSSL name, it returns "(NONE)". Where both exist, **stdname** should be the TLS name rather than the SSLv3 name.

SSL_CIPHER_get_bits() returns the number of secret bits used for **cipher**. If **cipher** is NULL, 0 is returned.

SSL_CIPHER_get_version() returns string which indicates the SSL/TLS protocol version that first defined the cipher. It returns "(NONE)" if **cipher** is NULL.

SSL_CIPHER_get_cipher_nid() returns the cipher NID corresponding to **c**. If there is no cipher (e.g. for cipher suites with no encryption) then **NID_undef** is returned.

SSL_CIPHER_get_digest_nid() returns the digest NID corresponding to the MAC used by **c** during record encryption/decryption. If there is no digest (e.g. for AEAD cipher suites) then **NID_undef** is returned.

SSL_CIPHER_get_handshake_digest() returns an EVP_MD for the digest used during the SSL/TLS handshake when using the SSL_CIPHER **c**. Note that this may be different to the digest used to calculate the MAC for encrypted records.

SSL_CIPHER_get_kx_nid() returns the key exchange NID corresponding to the method used by **c**. If there is no key exchange, then **NID_undef** is returned. If any appropriate key exchange algorithm can be used (as in the case of TLS 1.3 cipher suites) **NID_kx_any** is returned. Examples (not comprehensive):

NID_kx_rsa
NID_kx_ecdhe
NID_kx_dhe
NID_kx_psk

SSL_CIPHER_get_auth_nid() returns the authentication NID corresponding to the method used by **c**. If there is no authentication, then **NID_undef** is returned. If any appropriate authentication algorithm can be used (as in the case of TLS 1.3 cipher suites) **NID_auth_any** is returned. Examples (not comprehensive):

NID_auth_rsa
NID_auth_ecdsa
NID_auth_psk

SSL_CIPHER_is_aead() returns 1 if the cipher **c** is AEAD (e.g. GCM or ChaCha20/Poly1305), and 0 if it is not AEAD.

SSL_CIPHER_find() returns a **SSL_CIPHER** structure which has the cipher ID stored in **ptr**. The **ptr** parameter is a two element array of **char**, which stores the two-byte TLS cipher ID (as allocated by IANA) in network byte order. This parameter is usually retrieved from a TLS packet by using functions like **SSL_client_hello_get0_ciphers**(3). **SSL_CIPHER_find()** returns NULL if an error occurs or the

indicated cipher is not found.

SSL_CIPHER_get_id() returns the OpenSSL-specific ID of the given cipher **c**. That ID is not the same as the IANA-specific ID.

SSL_CIPHER_get_protocol_id() returns the two-byte ID used in the TLS protocol of the given cipher **c**.

SSL_CIPHER_description() returns a textual description of the cipher used into the buffer **buf** of length **len** provided. If **buf** is provided, it must be at least 128 bytes, otherwise a buffer will be allocated using **OPENSSL_malloc()**. If the provided buffer is too small, or the allocation fails, **NULL** is returned.

The string returned by **SSL_CIPHER_description()** consists of several fields separated by whitespace:

<ciphername>
: Textual representation of the cipher name.

<protocol version>
: The minimum protocol version that the ciphersuite supports, such as **TLSv1.2**. Note that this is not always the same as the protocol version in which the ciphersuite was first defined because some ciphersuites are backwards compatible with earlier protocol versions.

Kx=<key exchange>
: Key exchange method such as **RSA**, **ECDHE**, etc.

Au=<authentication>
: Authentication method such as **RSA**, **None**, etc.. None is the representation of anonymous ciphers.

Enc=<symmetric encryption method>
: Encryption method, with number of secret bits, such as **AESGCM(128)**.

Mac=<message authentication code>
: Message digest, such as **SHA256**.

Some examples for the output of **SSL_CIPHER_description()**:

```
ECDHE-RSA-AES256-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
RSA-PSK-AES256-CBC-SHA384 TLSv1.0 Kx=RSAPSK   Au=RSA  Enc=AES(256)  Mac=SHA384
```

RETURN VALUES

SSL_CIPHER_get_name(), **SSL_CIPHER_standard_name()**, **OPENSSL_cipher_name()**, **SSL_CIPHER_get_version()** and **SSL_CIPHER_description()** return the corresponding value in a NUL-terminated string for a specific cipher or "(NONE)" if the cipher is not found.

SSL_CIPHER_get_bits() returns a positive integer representing the number of secret bits or 0 if an error occurred.

SSL_CIPHER_get_cipher_nid(), **SSL_CIPHER_get_digest_nid()**, **SSL_CIPHER_get_kx_nid()** and **SSL_CIPHER_get_auth_nid()** return the NID value or **NID_undef** if an error occurred.

SSL_CIPHER_get_handshake_digest() returns a valid **EVP_MD** structure or NULL if an error occurred.

SSL_CIPHER_is_aead() returns 1 if the cipher is AEAD or 0 otherwise.

SSL_CIPHER_find() returns a valid **SSL_CIPHER** structure or NULL if an error occurred.

SSL_CIPHER_get_id() returns a 4-byte integer representing the OpenSSL-specific ID.

SSL_CIPHER_get_protocol_id() returns a 2-byte integer representing the TLS protocol-specific ID.

SEE ALSO

ssl(7), **SSL_get_current_cipher**(3), **SSL_get_ciphers**(3), **openssl-ciphers**(1)

HISTORY

The **SSL_CIPHER_get_version()** function was updated to always return the correct protocol string in OpenSSL 1.1.0.

The **SSL_CIPHER_description()** function was changed to return **NULL** on error, rather than a fixed string, in OpenSSL 1.1.0.

The **SSL_CIPHER_get_handshake_digest()** function was added in OpenSSL 1.1.1.

The **SSL_CIPHER_standard_name()** function was globally available in OpenSSL 1.1.1.
Before OpenSSL 1.1.1, tracing (**enable-ssl-trace** argument to Configure) was required to enable this function.

The **OPENSSL_cipher_name()** function was added in OpenSSL 1.1.1.

COPYRIGHT