SSL Ciphers

Ciphers

With curl's options `CURLOPT_SSL_CIPHER_LIST` and `--ciphers` users can control which ciphers to consider when negotiating TLS connections.

TLS 1.3 ciphers are supported since curl 7.61 for OpenSSL 1.1.1+ with options`CURLOPT_TLS13_CIPHERS` and `--tls13-ciphers` . If you are using a different SSL backend you can try setting TLS 1.3 cipher suites by using the respective regular cipher option.

The names of the known ciphers differ depending on which TLS backend that libcurl was built to use. This is an attempt to list known cipher names.

OpenSSL

(based on OpenSSL docs)

When specifying multiple cipher names, separate them with colon (`:`).

SSL3 cipher suites

`NULL-MD5 NULL-SHA RC4-MD5 RC4-SHA IDEA-CBC-SHA DES-CBC3-SHA DH-DSS-DES-CBC3-SHA DH-RSA-DES-CBC3-SHA DHE-DSS-DES-CBC3-SHA DHE-RSA-DES-CBC3-SHA ADH-RC4-MD5 ADH-DES-CBC3-SHA`

TLS v1.0 cipher suites

`NULL-MD5 NULL-SHA RC4-MD5 RC4-SHA IDEA-CBC-SHA DES-CBC3-SHA DHE-DSS-DES-CBC3-SHA DHE-RSA-DES-CBC3-SHA ADH-RC4-MD5 ADH-DES-CBC3-SHA`

AES ciphersuites from RFC3268, extending TLS v1.0

`AES128-SHA AES256-SHA DH-DSS-AES128-SHA DH-DSS-AES256-SHA DH-RSA-AES128-SHA DH-RSA-AES256-SHA DHE-DSS-AES128-SHA DHE-DSS-AES256-SHA DHE-RSA-AES128-SHA DHE-RSA-AES256-SHAADH-AES128-SHA ADH-AES256-SHA`

SEED ciphersuites from RFC4162, extending TLS v1.0

`SEED-SHA DH-DSS-SEED-SHA DH-RSA-SEED-SHA DHE-DSS-SEED-SHA DHE-RSA-SEED-SHA ADH-SEED-SHA`

GOST ciphersuites, extending TLS v1.0

`GOST94-GOST89-GOST89` `GOST2001-GOST89-GOST89``GOST94-NULL-GOST94` `GOST2001-NULL-GOST94`

Elliptic curve cipher suites

`ECDHE-RSA-NULL-SHA` `ECDHE-RSA-RC4-SHA` `ECDHE-RSA-DES-CBC3-SHA` `ECDHE-RSA-AES128-SHA` `ECDHE-RSA-AES256-SHA` `ECDHE-ECDSA-NULL-SHA` `ECDHE-ECDSA-RC4-SHA` `ECDHE-ECDSA-DES-CBC3-SHA` `ECDHE-ECDSA-AES128-SHA` `ECDHE-ECDSA-AES256-SHA``AECDH-NULL-SHA` `AECDH-RC4-SHA` `AECDH-DES-CBC3-SHA` `AECDH-AES128-SHA` `AECDH-AES256-SHA`

TLS v1.2 cipher suites

`NULL-SHA256` `AES128-SHA256` `AES256-SHA256``AES128-GCM-SHA256` `AES256-GCM-SHA384` `DH-RSA-AES128-SHA256` `DH-RSA-AES256-SHA256` `DH-RSA-AES128-GCM-SHA256` `DH-RSA-AES256-GCM-SHA384``DH-DSS-AES128-SHA256` `DH-DSS-AES256-SHA256` `DH-DSS-AES128-GCM-SHA256` `DH-DSS-AES256-GCM-SHA384` `DHE-RSA-AES128-SHA256` `DHE-RSA-AES256-SHA256` `DHE-RSA-AES128-GCM-SHA256` `DHE-RSA-AES256-GCM-SHA384` `DHE-DSS-AES128-SHA256` `DHE-DSS-AES256-SHA256` `DHE-DSS-AES128-GCM-SHA256``DHE-DSS-AES256-GCM-SHA384` `ECDHE-RSA-AES128-SHA256` `ECDHE-RSA-AES256-SHA384` `ECDHE-RSA-AES128-GCM-SHA256` `ECDHE-RSA-AES256-GCM-SHA384` `ECDHE-ECDSA-AES128-SHA256` `ECDHE-ECDSA-AES256-SHA384` `ECDHE-ECDSA-AES128-GCM-SHA256``ECDHE-ECDSA-AES256-GCM-SHA384` `ADH-AES128-SHA256` `ADH-AES256-SHA256` `ADH-AES128-GCM-SHA256` `ADH-AES256-GCM-SHA384` `AES128-CCM``AES256-CCM` `DHE-RSA-AES128-CCM` `DHE-RSA-AES256-CCM` `AES128-CCM8` `AES256-CCM8` `DHE-RSA-AES128-CCM8` `DHE-RSA-AES256-CCM8` `ECDHE-ECDSA-AES128-CCM` `ECDHE-ECDSA-AES256-CCM` `ECDHE-ECDSA-AES128-CCM8` `ECDHE-ECDSA-AES256-CCM8`

Camellia HMAC-Based ciphersuites from RFC6367, extending TLS v1.2

`ECDHE-ECDSA-CAMELLIA128-SHA256` `ECDHE-ECDSA-CAMELLIA256-SHA384` `ECDHE-RSA-CAMELLIA128-SHA256` `ECDHE-RSA-CAMELLIA256-SHA384`

TLS 1.3 cipher suites

(Note these ciphers are set with `CURLOPT_TLS13_CIPHERS` and `--tls13-ciphers`)

`TLS_AES_256_GCM_SHA384``TLS_CHACHA20_POLY1305_SHA256``TLS_AES_128_GCM_`

NSS

Totally insecure

`rc4` `rc4-md5` `rc4export` `rc2` `rc2export` `des``desede3`

SSL3/TLS cipher suites

`rsa_rc4_128_md5` `rsa_rc4_128_sha` `rsa_3des_sha``rsa_des_sha` `rsa_rc4_40_`

TLS 1.0 Exportable 56-bit Cipher Suites

`rsa_des_56_sha` `rsa_rc4_56_sha`

AES ciphers

`dhe_dss_aes_128_cbc_sha``dhe_dss_aes_256_cbc_sha``dhe_rsa_aes_128_cbc_`

ECC ciphers

`ecdh_ecdsa_null_sha` `ecdh_ecdsa_rc4_128_sha``ecdh_ecdsa_3des_sha` `ecdh_`

HMAC-SHA256 cipher suites

`rsa_null_sha_256` `rsa_aes_128_cbc_sha_256``rsa_aes_256_cbc_sha_256``dhe`

AES GCM cipher suites in RFC 5288 and RFC 5289

`rsa_aes_128_gcm_sha_256``dhe_rsa_aes_128_gcm_sha_256``dhe_dss_aes_128_`

cipher suites using SHA384

`rsa_aes_256_gcm_sha_384``dhe_rsa_aes_256_gcm_sha_384``dhe_dss_aes_256_`

chacha20-poly1305 cipher suites

`ecdhe_rsa_chacha20_poly1305_sha_256``ecdhe_ecdsa_chacha20_poly1305_`

TLS 1.3 cipher suites

`aes_128_gcm_sha_256` `aes_256_gcm_sha_384``chacha20_poly1305_sha_256`

GSKit

Ciphers are internally defined as numeric codes, but libcurl maps them to the following case-insensitive names.

SSL2 cipher suites (insecure: disabled by default)

`rc2-md5` `rc4-md5` `exp-rc2-md5` `exp-rc4-md5` `des-cbc-md5` `des-cbc3-md5`

SSL3 cipher suites

`null-md5` `null-sha` `rc4-md5` `rc4-sha` `exp-rc2-cbc-md5` `exp-rc4-md5` `exp-des-cbc-sha` `des-cbc3-sha`

TLS v1.0 cipher suites

null-md5 null-sha rc4-md5 rc4-sha exp-rc2-cbc-md5 exp-rc4-md5 exp-des-cbc-sha des-cbc3-sha aes128-sha aes256-sha

TLS v1.1 cipher suites

null-md5 null-sha rc4-md5 rc4-sha exp-des-cbc-sha des-cbc3-sha aes128-sha aes256-sha

TLS v1.2 cipher suites

null-md5 null-sha null-sha256 rc4-md5 rc4-shades-cbc3-sha aes128-sha aes256-sha aes128-sha256 aes256-sha256 aes128-gcm-sha256aes256-gcm-sha384

WolfSSL

RC4-SHA, RC4-MD5, DES-CBC3-SHA, AES128-SHA,AES256-SHA, NULL-SHA, NULL-SHA256, DHE-RSA-AES128-SHA, DHE-RSA-AES256-SHA, DHE-PSK-AES256-GCM-SHA384, DHE-PSK-AES128-GCM-SHA256,PSK-AES256-GCM-SHA384, PSK-AES128-GCM-SHA256,DHE-PSK-AES256-CBC-SHA384, DHE-PSK-AES128-CBC-SHA256, PSK-AES256-CBC-SHA384, PSK-AES128-CBC-SHA256, PSK-AES128-CBC-SHA, PSK-AES256-CBC-SHA, DHE-PSK-AES128-CCM, DHE-PSK-AES256-CCM, PSK-AES128-CCM, PSK-AES256-CCM,PSK-AES128-CCM-8, PSK-AES256-CCM-8, DHE-PSK-NULL-SHA384, DHE-PSK-NULL-SHA256, PSK-NULL-SHA384, PSK-NULL-SHA256, PSK-NULL-SHA, HC128-MD5, HC128-SHA, HC128-B2B256, AES128-B2B256,AES256-B2B256, RABBIT-SHA, NTRU-RC4-SHA, NTRU-DES-CBC3-SHA, NTRU-AES128-SHA, NTRU-AES256-SHA, AES128-CCM-8, AES256-CCM-8, ECDHE-ECDSA-AES128-CCM, ECDHE-ECDSA-AES128-CCM-8, ECDHE-ECDSA-AES256-CCM-8, ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES256-SHA, ECDHE-ECDSA-AES128-SHA,ECDHE-ECDSA-AES256-SHA, ECDHE-RSA-RC4-SHA,ECDHE-RSA-DES-CBC3-SHA, ECDHE-ECDSA-RC4-SHA,ECDHE-ECDSA-DES-CBC3-SHA, AES128-SHA256,AES256-SHA256, DHE-RSA-AES128-SHA256, DHE-RSA-AES256-SHA256, ECDH-RSA-AES128-SHA, ECDH-RSA-AES256-SHA, ECDH-ECDSA-AES128-SHA, ECDH-ECDSA-AES256-SHA, ECDH-RSA-RC4-SHA, ECDH-RSA-DES-CBC3-SHA, ECDH-ECDSA-RC4-SHA, ECDH-ECDSA-DES-CBC3-SHA, AES128-GCM-SHA256, AES256-GCM-SHA384, DHE-RSA-AES128-GCM-SHA256, DHE-RSA-AES256-GCM-SHA384, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-RSA-AES256-GCM-SHA384, ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES256-GCM-SHA384, ECDH-RSA-AES128-GCM-SHA256, ECDH-RSA-AES256-GCM-SHA384, ECDH-ECDSA-AES128-GCM-SHA256, ECDH-ECDSA-AES256-GCM-SHA384,CAMELLIA128-SHA, DHE-RSA-CAMELLIA128-SHA,CAMELLIA256-SHA, DHE-RSA-CAMELLIA256-SHA,CAMELLIA128-SHA256, DHE-RSA-CAMELLIA128-SHA256, CAMELLIA256-SHA256, DHE-RSA-CAMELLIA256-SHA256, ECDHE-RSA-AES128-SHA256,ECDHE-ECDSA-AES128-SHA256, ECDH-RSA-AES128-SHA256, ECDH-ECDSA-AES128-

`SHA256`, `ECDHE-RSA-AES256-SHA384`, `ECDHE-ECDSA-AES256-SHA384`,`ECDH-RSA-AES256-SHA384`, `ECDH-ECDSA-AES256-SHA384`, `ECDHE-RSA-CHACHA20-POLY1305`, `ECDHE-ECDSA-CHACHA20-POLY1305`, `DHE-RSA-CHACHA20-POLY1305`, `ECDHE-RSA-CHACHA20-POLY1305-OLD`,`ECDHE-ECDSA-CHACHA20-POLY1305-OLD`, `DHE-RSA-CHACHA20-POLY1305-OLD`, `ADH-AES128-SHA`, `QSH`,`RENEGOTIATION-INFO`, `IDEA-CBC-SHA`, `ECDHE-ECDSA-NULL-SHA`, `ECDHE-PSK-NULL-SHA256`, `ECDHE-PSK-AES128-CBC-SHA256`, `PSK-CHACHA20-POLY1305`,`ECDHE-PSK-CHACHA20-POLY1305`, `DHE-PSK-CHACHA20-POLY1305`, `EDH-RSA-DES-CBC3-SHA`,

Schannel

Schannel allows the enabling and disabling of encryption algorithms, but not specific ciphersuites. They are defined by Microsoft.

There is also the case that the selected algorithm is not supported by the protocol or does not match the ciphers offered by the server during the SSL negotiation. In this case curl will return error`CURLE_SSL_CONNECT_ERROR (35) SEC_E_ALGORITHM_MISMATCH` and the request will fail.

`CALG_MD2`, `CALG_MD4`, `CALG_MD5`, `CALG_SHA`,`CALG_SHA1`, `CALG_MAC`, `CALG_RSA_SI`