

**NAME**

CURLOPT\_PINNEDPUBLICKEY – set pinned public key

**SYNOPSIS**

```
#include <curl/curl.h>
```

```
CURLcode curl_easy_setopt(CURL *handle, CURLOPT_PINNEDPUBLICKEY, char *pinnedpubkey);
```

**DESCRIPTION**

Pass a pointer to a zero terminated string as parameter. The string can be the file name of your pinned public key. The file format expected is "PEM" or "DER". The string can also be any number of base64 encoded sha256 hashes preceded by "sha256//" and separated by ";

When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key provided to this option, curl will abort the connection before sending or receiving any data.

On mismatch, *CURLE\_SSL\_PINNEDPUBKEYNOTMATCH* is returned.

**DEFAULT**

NULL

**PROTOCOLS**

All TLS based protocols: HTTPS, FTPS, IMAPS, POP3, SMTPS etc.

**EXAMPLE**

```
CURL *curl = curl_easy_init();
if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");
    curl_easy_setopt(curl, CURLOPT_PINNEDPUBLICKEY, "/etc/publickey.der");
    /* OR
    curl_easy_setopt(curl, CURLOPT_PINNEDPUBLICKEY, "sha256//YhKJKSzoTt2b5FP18fvpHo7fJYqQCjAa3HWY3t
    */

    /* Perform the request */
    curl_easy_perform(curl);
}
```

**PUBLIC KEY EXTRACTION**

If you do not have the server's public key file you can extract it from the server's certificate.

# retrieve the server's certificate if you don't already have it

#

# be sure to examine the certificate to see if it is what you expected

#

# Windows-specific:

# - Use NUL instead of /dev/null.

# - OpenSSL may wait for input instead of disconnecting. Hit enter.

# - If you don't have sed, then just copy the certificate into a file:

# Lines from -----BEGIN CERTIFICATE----- to -----END CERTIFICATE-----.

#

```
openssl s_client -servername www.example.com -connect www.example.com:443 < /dev/null | sed -n "/-----BEGIN/-----"
```

# extract public key in pem format from certificate

```
openssl x509 -in www.example.com.pem -pubkey -noout > www.example.com.pubkey.pem
```

# convert public key from pem to der

```
openssl asn1parse -noout -inform pem -in www.example.com.pubkey.pem -out www.example.com.pubkey.der
```

```
# sha256 hash and base64 encode der to string for use
openssl dgst -sha256 -binary www.example.com.pubkey.der | openssl base64
The public key in PEM format contains a header, base64 data and a footer:
-----BEGIN PUBLIC KEY-----
[BASE 64 DATA]
-----END PUBLIC KEY-----
```

**AVAILABILITY**

Added in 7.39.0 for OpenSSL, GnuTLS and GSKit. Added in 7.43.0 for NSS and wolfSSL/CyaSSL. sha256 support added in 7.44.0 for OpenSSL, GnuTLS, NSS and wolfSSL/CyaSSL. Other SSL backends not supported.

**RETURN VALUE**

Returns `CURLE_OK` if TLS enabled, `CURLE_UNKNOWN_OPTION` if not, or `CURLE_OUT_OF_MEMORY` if there was insufficient heap space.

**SEE ALSO**

**CURLOPT\_SSL\_VERIFYPEER(3), CURLOPT\_SSL\_VERIFYHOST(3), CURLOPT\_CAINFO(3), CURLOPT\_CAPATH(3),**