FTPS

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/FTPS#mw-head)[Jump to search](https://en.wikipedia.org/wiki/FTPS#searchInput)

*This article is about FTP over SSL. For FTP Software, the defunct network software company, see*[*FTP Software*](https://en.wikipedia.org/wiki/FTP_Software)*.*

**FTPS** (also known **FTP-SSL**, and *FTP Secure*) is an extension to the commonly used [File Transfer Protocol](https://en.wikipedia.org/wiki/File_Transfer_Protocol) (FTP) that adds support for the [Transport Layer Security](https://en.wikipedia.org/wiki/Transport_Layer_Security) (TLS) and, formerly, the [Secure Sockets Layer](https://en.wikipedia.org/wiki/Secure_Sockets_Layer) (SSL, which is now prohibited by [RFC7568](https://tools.ietf.org/html/rfc7568)) cryptographic protocols.

FTPS should not be confused with the [SSH File Transfer Protocol](https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol) (SFTP), a secure file transfer subsystem for the [Secure Shell](https://en.wikipedia.org/wiki/Secure_Shell) (SSH) protocol with which it is not compatible. It is also different from [FTP over SSH](https://en.wikipedia.org/wiki/FTP_over_SSH), which is the practice of tunneling FTP through an SSH connection.



**Contents**

* [1Background](https://en.wikipedia.org/wiki/FTPS#Background)
* [2Methods of invoking security](https://en.wikipedia.org/wiki/FTPS#Methods_of_invoking_security)
  + [2.1Implicit](https://en.wikipedia.org/wiki/FTPS#Implicit)
  + [2.2Explicit](https://en.wikipedia.org/wiki/FTPS#Explicit)
* [3Transport Layer Security (TLS)/Secure Socket Layer (SSL)](https://en.wikipedia.org/wiki/FTPS#Transport_Layer_Security_(TLS)/Secure_Socket_Layer_(SSL))
  + [3.1General support](https://en.wikipedia.org/wiki/FTPS#General_support)
  + [3.2Scope of use](https://en.wikipedia.org/wiki/FTPS#Scope_of_use)
    - [3.2.1Secure command channel](https://en.wikipedia.org/wiki/FTPS#Secure_command_channel)
    - [3.2.2Secure data channel](https://en.wikipedia.org/wiki/FTPS#Secure_data_channel)
    - [3.2.3Reasons to disable encryption](https://en.wikipedia.org/wiki/FTPS#Reasons_to_disable_encryption)
    - [3.2.4SSL certificates](https://en.wikipedia.org/wiki/FTPS#SSL_certificates)
* [4Firewall incompatibilities](https://en.wikipedia.org/wiki/FTPS#Firewall_incompatibilities)
* [5See also](https://en.wikipedia.org/wiki/FTPS#See_also)
* [6Notes](https://en.wikipedia.org/wiki/FTPS#Notes)
* [7External links](https://en.wikipedia.org/wiki/FTPS#External_links)

Background[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=1)]

The File Transfer Protocol was drafted in 1971 for use with the scientific and research network, [ARPANET](https://en.wikipedia.org/wiki/ARPANET).[[1]](https://en.wikipedia.org/wiki/FTPS#cite_note-rfc265-1) Access to the ARPANET during this time was limited to a small number of military sites and universities and a narrow community of users who could operate without data security and privacy requirements within the protocol.

As the ARPANET gave way to the [NSFnet](https://en.wikipedia.org/wiki/NSFnet" \o "NSFnet) and then the Internet, a broader population potentially had access to the data as it traversed increasingly longer paths from client to server. The opportunity for unauthorized third parties to [eavesdrop](https://en.wikipedia.org/wiki/Eavesdropping) on data transmissions increased proportionally.

In 1994, the Internet browser company [Netscape](https://en.wikipedia.org/wiki/Netscape) developed and released the [application layer](https://en.wikipedia.org/wiki/Application_layer) wrapper, [Secure Sockets Layer](https://en.wikipedia.org/wiki/Secure_Sockets_Layer).[[2]](https://en.wikipedia.org/wiki/FTPS#cite_note-ssl2draft-2) This protocol enabled applications to communicate across a network in a private and secure fashion, discouraging eavesdropping, tampering, and message forgery. While it could add security to any protocol that uses reliable connections, such as [TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol), it was most commonly used by Netscape with HTTP to form HTTPS.

The SSL protocol was eventually applied to FTP, with a draft [Request for Comments](https://en.wikipedia.org/wiki/Request_for_Comments) (RFC) published in late 1996.[[3]](https://en.wikipedia.org/wiki/FTPS#cite_note-ftpsdraft-3) An official [IANA](https://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority) port was registered shortly thereafter. However, the RFC was not finalized until 2005.[[4]](https://en.wikipedia.org/wiki/FTPS#cite_note-rfc4217-4)

Methods of invoking security[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=2)]

Two separate methods were developed to invoke client security for use with FTP clients: *Implicit* and *Explicit*. While the implicit method requires that a Transport Layer Security is established from the beginning of the connection, which in turn breaks the compatibility with non-FTPS-aware clients and servers, the explicit method uses standard FTP protocol commands and replies in order to upgrade a plain text connection to an encrypted one, allowing a single control port to be used for serving both FTPS-aware and non-FTPS-aware clients.

**Implicit**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=3)]

Negotiation is not supported with implicit FTPS configurations. A client is immediately expected to challenge the FTPS server with a [TLS](https://en.wikipedia.org/wiki/Transport_Layer_Security) *ClientHello* message. If such a message is not received by the FTPS server, the server should drop the connection.

In order to maintain compatibility with existing non-FTPS-aware clients, implicit FTPS was expected to listen on the IANA well known port 990/TCP for the FTPS control channel, and port 989/TCP for the FTPS data channel.[[5]](https://en.wikipedia.org/wiki/FTPS#cite_note-5) This allowed administrators to retain legacy-compatible services on the original 21/TCP FTP control channel.

Note that implicit negotiation was not defined in RFC 4217. As such, it is considered an earlier, deprecated method of negotiating TLS/SSL for FTP.[[6]](https://en.wikipedia.org/wiki/FTPS#cite_note-draft-murray-auth-ftp-ssl-07-6)

**Explicit**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=4)]

In explicit mode (also known as FTPES), an FTPS client must "explicitly request" security from an FTPS server and then step up to a mutually agreed encryption method. If a client does not request security, the FTPS server can either allow the client to continue in insecure mode or refuse the connection.

The mechanism for negotiating authentication and security with FTP was added under [RFC 2228](https://tools.ietf.org/html/rfc2228), which included the new FTP command AUTH. While this RFC does not explicitly define any required security mechanisms, e.g. SSL or TLS, it does require the FTPS client to challenge the FTPS server with a mutually known mechanism. If the FTPS client challenges the FTPS server with an unknown security mechanism, the FTPS server will respond to the AUTH command with error code *504 (not supported)*. Clients may determine which mechanisms are supported by querying the FTPS server with the FEAT command, although servers are not necessarily required to be honest in disclosing what levels of security they support. Common methods of invoking FTPS security included AUTH TLS and AUTH SSL.

The explicit method is defined in [RFC 4217](https://tools.ietf.org/html/rfc4217). In the later versions of the document, FTPS compliance required that clients always negotiate using the AUTH TLS method.

Transport Layer Security (TLS)/Secure Socket Layer (SSL)[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=5)]

*Main article:*[*Transport Layer Security*](https://en.wikipedia.org/wiki/Transport_Layer_Security)

**General support**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=6)]

FTPS includes full support for the TLS and SSL cryptographic protocols, including the use of server-side [public key authentication certificates](https://en.wikipedia.org/wiki/Public_key_certificate) and client-side authorization certificates. It also supports compatible ciphers, including [AES](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard), [RC4](https://en.wikipedia.org/wiki/RC4_(cipher)), [RC2](https://en.wikipedia.org/wiki/RC2), [Triple DES](https://en.wikipedia.org/wiki/Triple_DES), and [DES](https://en.wikipedia.org/wiki/Data_Encryption_Standard). It further supports hash functions [SHA](https://en.wikipedia.org/wiki/SHA_family), [MD5](https://en.wikipedia.org/wiki/MD5), [MD4](https://en.wikipedia.org/wiki/MD4), and [MD2](https://en.wikipedia.org/wiki/MD2_(cryptography)).

**Scope of use**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=7)]

In implicit mode, the entire FTPS session is encrypted. Explicit mode differs in that the client has full control over what areas of the connection are to be encrypted. Enabling and disabling of encryption for the FTPS control channel and FTPS data channel can occur at any time. The only restriction comes from the FTPS server, which has the ability to deny commands based on server encryption policy.

**Secure command channel**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=8)]

The secure command channel mode can be entered through the issue of either the AUTH TLS or AUTH SSL commands. After such time, all command control between the FTPS client and server are assumed to be encrypted. It is generally advised to enter such a state prior to user authentication and authorization in order to avoid the eavesdropping of user name and password data by third parties.

**Secure data channel**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=9)]

The secure data channel can be entered through the issue of the PROT command. It is *not* enabled by default when the AUTH TLS command is issued. After such time, all data channel communication between the FTPS client and server is assumed to be encrypted.

The FTPS client may exit the secure data channel mode at any time by issuing a CDC (clear data channel) command.

**Reasons to disable encryption**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=10)]

It may not be advantageous to use data channel encryption when performing transfers under the following scenarios:

* Files being transferred are of a non-sensitive nature, making encryption unnecessary,
* Files being transferred are already encrypted at the file level or are passing over an encrypted [VPN](https://en.wikipedia.org/wiki/VPN), making encryption redundant,
* Available TLS or SSL encryption modes do not meet desired level of encryption. This is common with older FTPS clients or servers that may have been [limited to 40-bit SSL](https://en.wikipedia.org/wiki/Transport_Layer_Security#Government-imposed_protocol_limitations) due to previous United States high-encryption export laws.

It may not be advantageous to use control channel encryption under the following scenarios:

* Use of FTPS when the client or server reside behind a [network firewall](https://en.wikipedia.org/wiki/Network_firewall) or [network address translation](https://en.wikipedia.org/wiki/Network_address_translation) (NAT) device. (See [Firewall Incompatibilities](https://en.wikipedia.org/wiki/FTPS#Firewall_incompatibilities) below.)
* Repeated use of AUTH and CCC/CDC commands by anonymous FTP clients within the same session. Such behavior can be used as a resource-based denial of service attack as the TLS/SSL session must be regenerated each time, using server processor time.

**SSL certificates**[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=11)]

Much like [HTTPS](https://en.wikipedia.org/wiki/HTTPS), FTPS servers must provide a [public key certificate](https://en.wikipedia.org/wiki/Public_key_certificate). These certificates can be requested and created using tools such as [OpenSSL](https://en.wikipedia.org/wiki/OpenSSL).

When these certificates are signed by a trusted [certificate authority](https://en.wikipedia.org/wiki/Certificate_authority), this provides assurance that the client is connected to the requested server, avoiding a [man-in-the-middle attack](https://en.wikipedia.org/wiki/Man-in-the-middle_attack). If the certificate is not signed by a trusted CA (a [self-signed certificate](https://en.wikipedia.org/wiki/Self-signed_certificate)), the FTPS client may generate a warning stating that the certificate is not valid. The client can choose to accept the certificate or reject the connection.

This is in contrast to the [SSH File Transfer Protocol](https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol) (SFTP), which does not present signed certificates, but instead relies on [Out-of-band authentication](https://en.wikipedia.org/wiki/Out-of-band_authentication) of public keys.

Firewall incompatibilities[[edit](https://en.wikipedia.org/w/index.php?title=FTPS&action=edit&section=12)]

Because FTP uses a dynamic secondary port (for data channels), many [firewalls](https://en.wikipedia.org/wiki/Firewall_(networking)) were designed to snoop FTP protocol control messages in order to determine which secondary data connections they need to allow. However, if the FTP control connection is encrypted using TLS/SSL, the firewall cannot determine the TCP port number of a data connection negotiated between the client and FTP server. Therefore, in many firewalled networks, an FTPS deployment will fail when an unencrypted FTP deployment will work. This problem can be solved with the use of a limited range of ports for data and configuring the firewall to open these ports.