[MS-SSTP]:
Secure Socket Tunneling Protocol (SSTP) Specification

Revision Summary

Date	Revision History	Revision Class	Comments
02/22/2007	0.01		MCPP Milestone 3 Initial Availability
04/03/2007	0.01		MCPP Milestone Longhorn Initial Availability
06/01/2007	1.0	Major	Updated and revised the technical content.
07/03/2007	1.1	Minor	Updated technical content.
07/20/2007	1.1.1	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
08/10/2007	1.1.2	Editorial	Revised and edited the technical content.
09/28/2007	1.1.3	Editorial	Revised and edited the technical content.
10/23/2007	1.1.4	Editorial	Revised and edited the technical content.
11/30/2007	1.1.5	Editorial	Revised and edited the technical content.
01/25/2008	1.1.6	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction 6
1.1 Glossary 6
1.2 References 6
1.2.1 Normative References 6
1.2.2 Informative References 7
1.3 Protocol Overview (Synopsis) 7
1.4 Relationship to Other Protocols 9
1.5 Prerequisites/Preconditions 9
1.6 Applicability Statement 9
1.7 Versioning and Capability Negotiation 9
1.8 Vendor-Extensible Fields 10
1.9 Standards Assignments 10

2 Messages 11
2.1 Transport 11
2.2 Message Syntax 11
2.2.1 SSTP Packet 11
2.2.2 SSTP Control Packet 12
2.2.3 SSTP Data Packet 14
2.2.4 SSTP Attributes 15
2.2.5 Encapsulated Protocol ID Attribute 16
2.2.6 Crypto Binding Request Attribute 17
2.2.7 Crypto Binding Attribute 19
2.2.8 Status Info Attribute 21
2.2.9 Call Connect Request Message 25
2.2.10 Call Connect Acknowledge Message 26
2.2.11 Call Connected Message 29
2.2.12 Call Connect Negative Acknowledgment Message 32
2.2.13 Call Abort and Call Disconnect Messages 36
2.2.14 Call Disconnect Acknowledge, Echo Request, and Echo Response Messages 37

3 Protocol Details 39
3.1 Client Details 39
3.1.1 Abstract Data Model 39
3.1.1.1 State Machine 39
3.1.1.1.1 Call Establishment 39
3.1.2 Timers 40
3.1.2.1 Negotiation Timer 41
3.1.3 Initialization 41
3.1.4 Higher-Layer Triggered Events 41
3.1.4.1 Establish Tunnel Event 41
3.1.4.2 Disconnect Tunnel Event 41
3.1.5 Message Processing Events and Sequencing Rules 42
3.1.5.1 Status and Error Handling 42
3.1.5.2 Crypto Binding 42
3.1.5.2.1 Input Data Used in the Crypto Binding HMAC-SHA1-160 Operation 43
3.1.5.2.2 Key Used in the Crypto Binding HMAC-SHA1-160 Operation 43
3.1.5.2.3 Input Data Used in the Crypto Binding HMAC-SHA256-256 Operation 44
3.1.5.2.4 Key Used in the Crypto Binding HMAC-SHA256-256 Operation 44
3.1.5.3 Packet Processing 46
3.1.5.3.1 General Packet Validation 46
3.1.5.3.2 Receiving a Call Connect Acknowledge Message 46

3.1.5.3.3 Receiving a Call Connect Negative Acknowledgment Message 46
3.1.5.3.4 Receiving a Call Abort Message 47
3.1.5.3.5 Receiving a Call Disconnect Message 47
3.1.5.3.6 Receiving a Call Disconnect Acknowledge Message 47
3.1.5.3.7 Receiving an Echo Request Message 47
3.1.5.3.8 Receiving an Echo Response Message 47
3.1.6 Timer Events 47
3.1.6.1 Negotiation Timer Processing 47
3.1.7 Other Local Events 47
3.2 Server Details 47
3.2.1 Abstract Data Model 47
3.2.1.1 State Machine 48
3.2.1.1.1 Call Establishment 48
3.2.2 Timers 49
3.2.2.1 Negotiation Timer 49
3.2.3 Initialization 49
3.2.4 Higher-Layer Triggered Events 49
3.2.5 Message Processing Events and Sequencing Rules 49
3.2.5.1 Status and Error Handling 49
3.2.5.2 Packet Processing 49
3.2.5.2.1 General Packet Validation 49
3.2.5.2.2 Receiving a Call Connect Request Message 49
3.2.5.2.3 Receiving a Call Connected Message 50
3.2.5.2.4 Receiving a Call Abort Message 50
3.2.5.2.5 Receiving a Call Disconnect Message 51
3.2.5.2.6 Receiving a Call Disconnect Acknowledge Message 51
3.2.5.2.7 Receiving an Echo Request Message 51
3.2.5.2.8 Receiving an Echo Response Message 51
3.2.6 Timer Events 51
3.2.6.1 Negotiation Timer Processing 51
3.2.7 Other Local Events 51
3.3 Common Details 51
3.3.1 Abstract Data Model 51
3.3.1.1 State Machine 51
3.3.1.1.1 State Machine Call Disconnect 52
3.3.1.1.2 State Machine Call Abort 52
3.3.2 Timers 53
3.3.2.1 Abort-Related Timers 53
3.3.2.2 Disconnect-Related Timers 53
3.3.2.3 Hello Timer 54
3.3.3 Initialization 54
3.3.4 Higher-Layer Triggered Events 54
3.3.5 Message Processing Events and Sequencing Rules 54
3.3.5.1 Status and Error Handling 54
3.3.5.2 SSTP Packet Processing 55
3.3.6 Timer Events 55
3.3.6.1 Abort Timer Processing 55
3.3.6.2 Disconnect Timer Processing 55
3.3.6.3 Hello Timer Processing 55
3.3.7 Other Local Events 55

4 Protocol Examples 56
4.1 HTTPS Layer Establishment 56
4.2 HTTP Layer Teardown 56
4.3 SSTP Layer Establishment 57

4.4 SSTP Layer Teardown 57
4.5 Handling HTTP Proxies 58
4.6 Crypto Binding 59

5 Security 65
5.1 Security Considerations for Implementers 65
5.2 Index of Security Parameters 65
5.3 Attack Scenarios 65
5.3.1 Unauthorized Client Connecting to an SSTP Server 65
5.3.2 Unauthorized SSTP Server Accepting Connections from a Genuine SSTP Client 66
5.3.3 Man in the Middle 67

6 Appendix A: Windows Behavior 70

7 Index 71

1 Introduction

This document describes the Microsoft Secure Socket Tunneling Protocol (SSTP), a mechanism to transport data-link layer (L2) frames on a Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) connection. The protocol currently supports only the Point-to-Point Protocol (PPP) link layer (for more information, see [RFC1661]).

This protocol has two main deployment modes:

- The **SSTP server** directly accepts the HTTPS connection.

  In this scenario, the SSTP server accepts the HTTPS connection, which is similar to a virtual private network (VPN) server positioned on the edge of a network. The Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificate is deployed on the SSTP server.

- The SSTP server is positioned behind an SSL/TLS load balancer.

  In this scenario, the SSTP server is positioned behind an SSL/TLS load balancer that terminates the SSL/TLS connections (and therefore, the SSL/TLS certificate is installed) and forwards the decrypted HTTP traffic to the SSTP server. There is an implicit relationship of trust between the load balancer (or trusted man-in-the-middle) and the SSTP server.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

State Machine

The following terms are specific to this document:

SSTP Client: A computer that implements the Secure Socket Tunneling Protocol (SSTP) and that initiates an SSTP connection to an **SSTP server** over TCP port 443.

SSTP Far End: An entity that has sent a Secure Socket Tunneling Protocol (SSTP) message that is currently being processed by an **SSTP peer** and to whom the response is sent by the **SSTP peer**.

SSTP Peer: An entity that processes a Secure Socket Tunneling Protocol (SSTP) message.

SSTP Server: An entity on a network that implements the Secure Socket Tunneling Protocol (SSTP) and that listens for SSTP connections over TCP port 443.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[RFC1945] Berners-Lee, T., Fielding, R., and Frystyk, H., "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996, http://www.ietf.org/rfc/rfc1945.txt

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2284] Blunk, L. and Vollbrecht, J., "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998, http://www.ietf.org/rfc/rfc2284.txt

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2716] Aboba, B. and Simon, D., "PPP EAP TLS Authentication Protocol", RFC 2716, October 1999, http://www.ietf.org/rfc/rfc2716.txt

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000, http://www.ietf.org/rfc/rfc2759.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC3079] Zorn, G., "Deriving Keys for Use with Microsoft Point-to-Point Encryption (MPPE)", RFC 3079, March 2001, http://www.ietf.org/rfc/rfc3079.txt

[RFC3174] Eastlake III, D. and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, http://www.ietf.org/rfc/rfc3174.txt

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and Levkowetz, H., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004, http://www.ietf.org/rfc/rfc3748.txt

[SHA256] National Institute of Standards and Technology, "FIPS 180-2, Secure Hash Standard (SHS)", August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[SSLPROXY] Luotonen, A., "Tunneling SSL Through a WWW Proxy", March 1997, http://tools.ietf.org/html/draft-luotonen-ssl-tunneling-03

1.2.2 Informative References

[MS-PEAP] Microsoft Corporation, "Protected Extensible Authentication Protocol (PEAP) Specification", January 2007.

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", RFC 1661, July 1994, http://www.ietf.org/rfc/rfc1661.txt

[RFC1750] Eastlake III, D., Crocker, S., and Schiller, J., "Randomness Recommendations for Security", RFC 1750, December 1994, http://www.ietf.org/rfc/rfc1750.txt

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005, http://www.ietf.org/rfc/rfc4306.txt

1.3 Protocol Overview (Synopsis)

This specification defines the Secure Socket Tunneling Protocol (SSTP). SSTP is a mechanism to encapsulate Point-to-Point Protocol (PPP) traffic over an HTTPS protocol, as specified in [RFC1945],

[RFC2616], and [RFC2818]. This protocol enables users to access a private network using HTTPS. The use of HTTPS enables traversal of most firewalls and Web proxies.

Many VPN services provide a way for mobile and home users to access the corporate network remotely by using the Point-to-Point Tunneling Protocol (PPTP) and the Layer Two Tunneling Protocol/Internet Protocol security (L2TP/IPsec). However, with the popularization of firewalls and Web proxies, many service providers (for example, hotels) do not allow the PPTP and L2TP/IPsec traffic. This results in users not receiving ubiquitous connectivity to their corporate networks. For example, generic routing encapsulation (GRE) port blocking by many Internet service providers (ISPs) is a common problem when using PPTP.

The Secure Socket Tunneling Protocol (SSTP) provides an encrypted tunnel by means of the SSL/TLS protocol. When a client establishes an SSTP-based VPN connection, it first establishes a TCP connection to the SSTP server over TCP port 443. SSL/TLS negotiation occurs over this TCP connection. The client then validates the server certificate it receives during the SSL authentication phase. If the server certificate is not valid or if it is not trusted by the client, the client terminates the connection.

After the successful negotiation of SSL/TLS, the client sends an HTTP request with content length encoding and a large content length on the SSL protected connection. The server sends back an HTTP response with status HTTP_STATUS_OK(200). The HTTPS connection is now established, and the client can send and receive SSTP Control Packets and SSTP Data Packets on this connection. HTTPS connection establishment when a Web proxy is present is specified in [SSLPROXY].

The Secure Socket Tunneling Protocol (SSTP) performs the following functions:

- Allowing delineation of PPP frames from the continuous stream of data that is sent by using HTTPS. For more information about PPP, see [RFC1661].
- Negotiation of parameters between two entities. Note that this feature is for future extensibility and not completely used.
- Extensible message format to support new parameters in the future.
- Security operations to prevent a man-in-the-middle attacker from relaying PPP frames inappropriately over SSTP.

SSTP Control Packets contain messages to negotiate parameters and to ensure that there is no untrusted man-in-the-middle. SSTP Data Packets contain PPP frames as payload.

In an SSTP-based VPN, the protocol layer negotiation occurs in the following order:

- The TCP connection is established to an SSTP server over TCP port 443.
- SSL/TLS negotiation is completed.
- HTTPS request-response is completed.
- SSTP negotiation begins.
- PPP negotiation is initiated, and PPP authentication is completed.
- SSTP negotiation is completed.
- PPP negotiation is completed.
- The connection enters a ready state for transportation of any network layer; for example, Internet Protocol (IP) packets.

The following encapsulation operations occur on the client:

- Application packets are encapsulated over any transport protocol (for example, TCP and UDP).
- Transport layer packets are encapsulated over a network protocol (for example, IP).
- Network layer packets are encapsulated over a PPP data-link layer.
- PPP packets are encapsulated over SSTP.
- SSTP Packets are encapsulated over SSL/TLS.
- SSL/TLS records are encapsulated over TCP.
- TCP packets are encapsulated over IP.
- IP packets are sent over any data-link layer (such as Ethernet or PPP). For more information about PPP, see [RFC1661].

On the server side, operations to remove the encapsulation occur in reverse order.

1.4 Relationship to Other Protocols

The Secure Socket Tunneling Protocol (SSTP) allows encapsulation of Point-to-Point Protocol (PPP) traffic over HTTPS, as specified in [RFC1945], [RFC2616], and [RFC2818]. For more information about PPP, see [RFC1661].

1.5 Prerequisites/Preconditions

The Secure Socket Tunneling Protocol (SSTP) supports encapsulation of Point-to-Point Protocol (PPP) traffic over HTTPS, as specified in [RFC1945], [RFC2616], and [RFC2818]. As a result, SSTP requires understanding and implementation of both PPP and HTTPS for operation. For more information about PPP, see [RFC1661].

SSTP also requires PPP to use an authentication protocol for security. SSTP requires a certificate that is configured on the server side prior to connection.

1.6 Applicability Statement

The Secure Socket Tunneling Protocol (SSTP) is useful for establishing VPN connections over public networks. If SSTP is run in a high-loss environment, TCP connections may encounter limited data transfer rates over SSTP.

1.7 Versioning and Capability Negotiation

The Secure Socket Tunneling Protocol (SSTP) contains version fields in all the messages. This functionality allows a server implementation to support multiple versions of SSTP and to determine what version is implemented on the client. The server then uses that version of protocol behavior for SSTP negotiation.

SSTP also allows for negotiation of SHA1 and SHA256 for hashing. The negotiation is as specified in sections 2.2.7, 2.2.6, 2.2.10, 3.1.5.2, and 3.1.5.3.2.

The server presents the hashing functions that it supports, and the client selects the hashing function that it also supports from this list. If both the client and server support both SHA1 and SHA256, the client prefers SHA256 over SHA1.

1.8 Vendor-Extensible Fields

The Secure Socket Tunneling Protocol (SSTP) has no vendor-extensible fields.

1.9 Standards Assignments

The Secure Socket Tunneling Protocol (SSTP) has not been assigned any standards from any accepted industry standards body.

2 Messages

The following sections specify how Secure Socket Tunneling Protocol (SSTP) messages are transported and message syntax.

2.1 Transport

The following diagram shows the SSTP protocol stack.

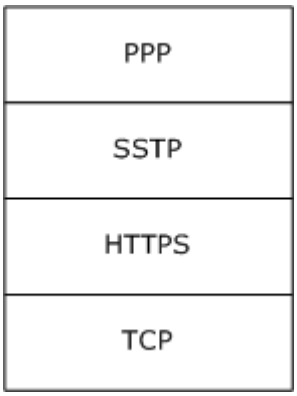


Figure 1: SSTP protocol stack

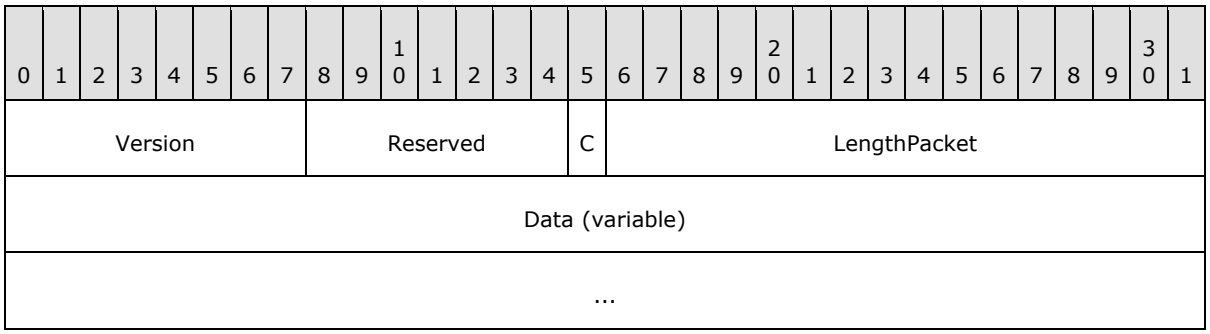
Because SSTP runs by means of an HTTPS connection, SSTP relies entirely on HTTPS for the reliable delivery of its messages. The **SSTP client** MUST authenticate the SSTP server by using HTTPS authentication. The SSTP server MAY<1> authenticate the SSTP client by using HTTPS client authentication. The SSTP server MUST authenticate the SSTP client by using PPP authentication. Therefore, PPP authentication is required even when the SSTP server authenticates the SSTP client by using HTTPS authentication. For more information about PPP, see [RFC1661].

2.2 Message Syntax

2.2.1 SSTP Packet

The following diagram shows the format of the SSTP packet that is sent on the HTTPS connection.

The fields of the header MUST be transmitted in byte order from left to right.



Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is used. The upper 4 bits are the MAJOR version, which MUST be 0x1, and the

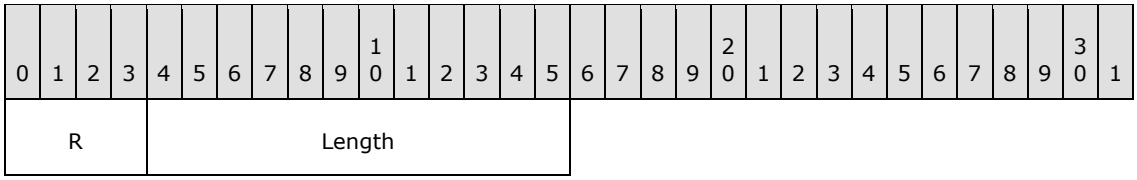
lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet (the data packet is used for sending a higher-layer payload). The value is 1 if it is a control packet and 0 if it is a data packet.

Name	Value
Data packet	0
Control packet	1

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

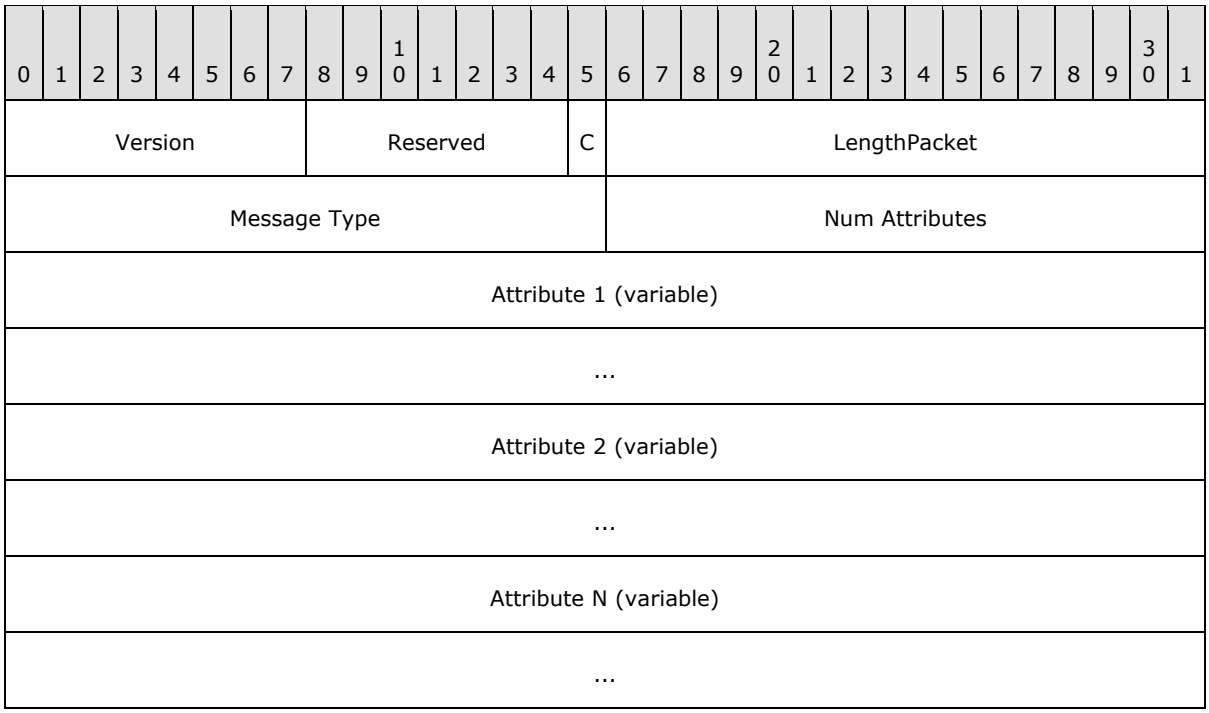
Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire SSTP packet, including the 4-byte SSTP header (that is: Version, 7-bit Reserved field, 1-bit C field, 4-bit R field, and 12-bit Length field).

Data (variable): A variable-length field. The length of this field is equal to the value of the field **Length**, minus 4. This field contains either the SSTP control message when field **C** is equal to 1 (see section 2.2.2), or the payload from a higher-layer protocol when field **C** is equal to 0 (see section 2.2.3). SSTP data packets carry PPP frames as payload (which includes PPP control frames as well as PPP data frames). For more information, see [RFC1661].

2.2.2 SSTP Control Packet

The SSTP control packet is a type of SSTP packet that is used by both the client and the server to send control messages to each other. The following diagram specifies the format that MUST be used for the SSTP control messages. Because this message is a type of SSTP packet, it follows the format of an SSTP packet (section 2.2.1).

The fields of the structure MUST be transmitted in network byte order from left to right.

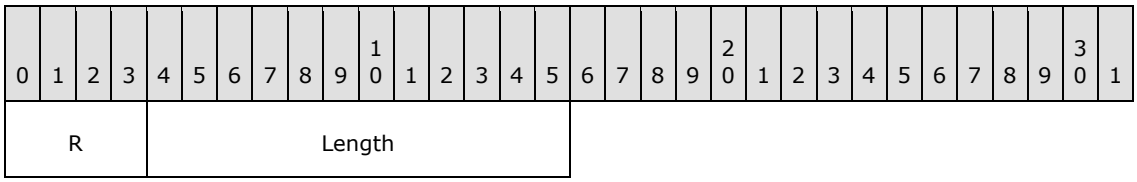


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is used. The upper 4 bits are the MAJOR version, which MUST be 0x1 and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be set to 1 for control packets.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire SSTP packet, including the 4-byte SSTP header (that is, Version, 7-bit Reserved field, 1-bit C field, 4-bit R field, and 12-bit Length field).

Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. It MUST be one of the following values.

Name	Value
SSTP_MSG_CALL_CONNECT_REQUEST	0x0001
SSTP_MSG_CALL_CONNECT_ACK	0x0002
SSTP_MSG_CALL_CONNECT_NAK	0x0003
SSTP_MSG_CALL_CONNECTED	0x0004
SSTP_MSG_CALL_ABORT	0x0005
SSTP_MSG_CALL_DISCONNECT	0x0006
SSTP_MSG_CALL_DISCONNECT_ACK	0x0007
SSTP_MSG_ECHO_REQUEST	0x0008
SSTP_MSG_ECHO_RESPONSE	0x0009

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message.

Attribute 1 (variable): MUST contain the first attribute.

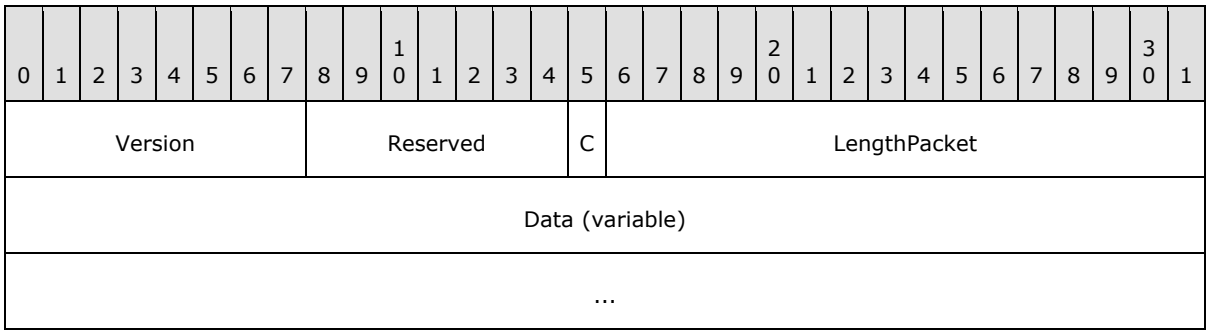
Attribute 2 (variable): MUST contain the second attribute.

Attribute N (variable): An ordered list of variable-sized attributes that compose an SSTP control message. Each attribute must follow the format as specified in section 2.2.4.

2.2.3 SSTP Data Packet

The SSTP data packet is a type of SSTP packet that is used by both the SSTP client and SSTP server to send a higher-layer payload (that is, a PPP frame) to each other. The following diagram specifies the format that MUST be used for the SSTP data packet. For more information, see [RFC1661].

The fields of the structure MUST be transmitted as bytes from left to right.

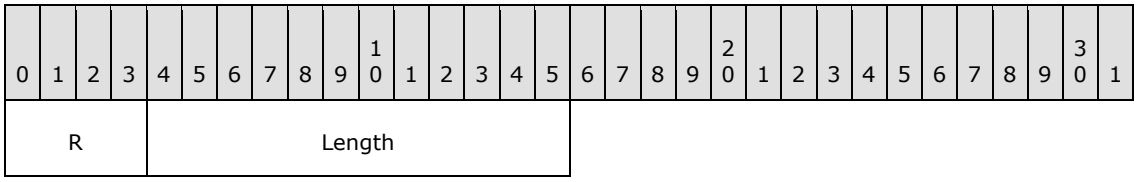


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP being used. The upper 4 bits are the MAJOR version, which MUST be 0x1, and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be set to 0 to indicate a data packet that carries higher-layer payloads.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

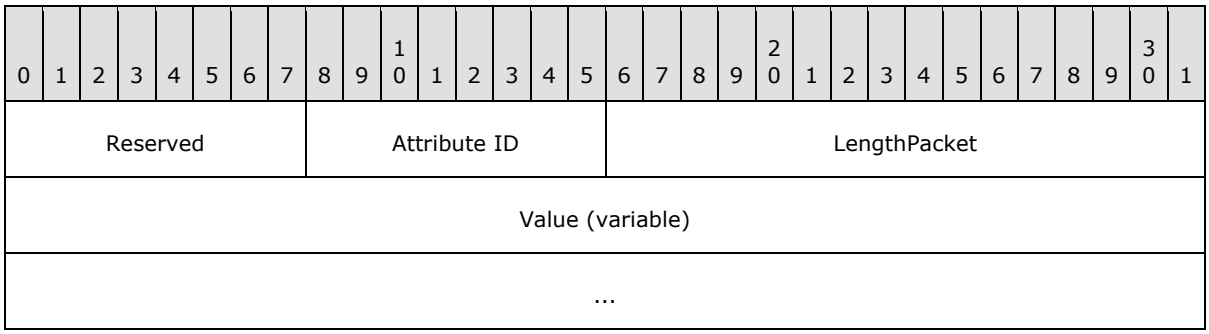
Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire SSTP packet, including the 4-byte SSTP header (that is, Version, 7-bit Reserved field, 1-bit C field, 4-bit R field, and 12-bit Length field).

Data (variable): A variable-length field that contains the higher-layer payload. The length of this field is equal to the value of the field **Length**, minus 4.

2.2.4 SSTP Attributes

The following diagram specifies the format that MUST be used for all SSTP attributes.

The fields of the structure MUST be transmitted as bytes from left to right.



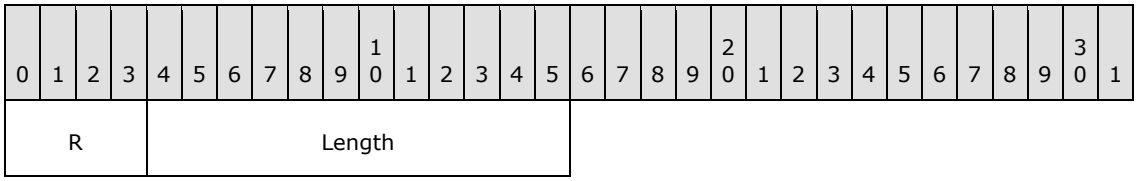
Reserved (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be one of the following.

Name	Value
SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID	0x01
SSTP_ATTRIB_STATUS_INFO	0x02
SSTP_ATTRIB_CRYPTO_BINDING	0x03

Name	Value
SSTP_ATTRIB_CRYPTO_BINDING_REQ	0x04

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire SSTP attribute, including the 4-byte attribute header (that is, 1-byte Reserved field, 1-byte Attribute ID, 4-bit R field, and 12-bit Length field).

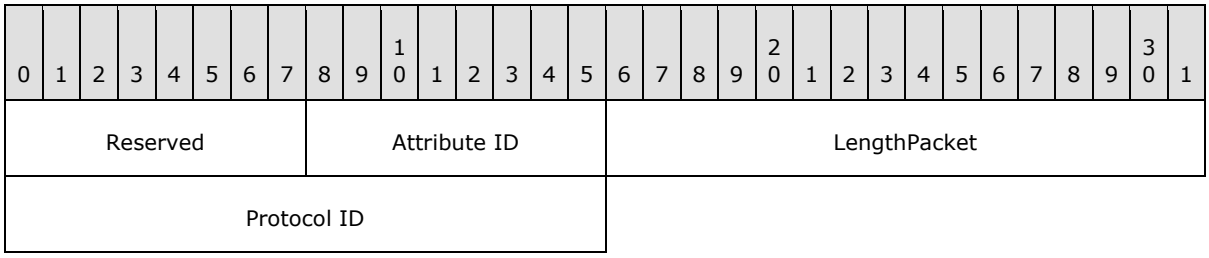
Value (variable): A variable-length field with length equal to field **Length** minus 4 that contains the attribute-specific data.

2.2.5 Encapsulated Protocol ID Attribute

The following diagram specifies the format that MUST be used for the Encapsulated Protocol ID attribute. This attribute is used to negotiate the higher-layer protocols that are supported by the client and server.

The client proposes the list of higher-layer protocols that it wants to send on the SSTP connection. If the server supports the protocols that are specified by the client, a negative acknowledge response is not sent. Otherwise, the server sends a negative acknowledge response with the list of protocols that it does not support.

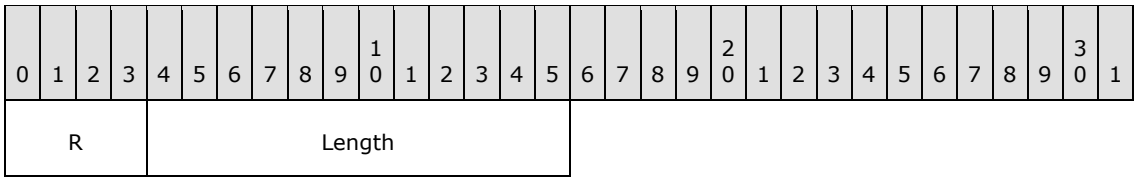
The fields of the structure MUST be transmitted as bytes from left to right.



Reserved (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x01 for the Encapsulated Protocol ID attribute.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that contains the value 0x006 for the Encapsulated Protocol ID attribute length.

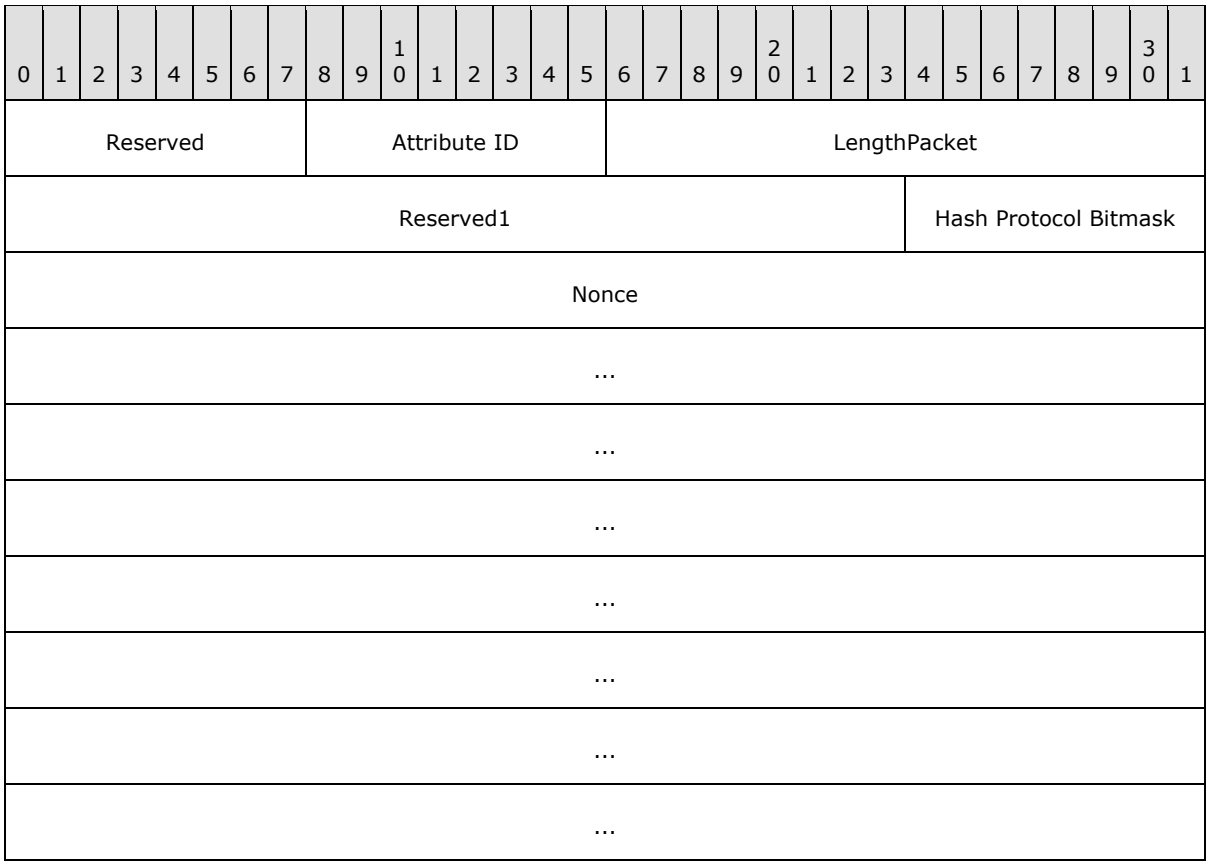
Protocol ID (2 bytes): A 2-byte field in network byte order that contains the following value. Note that SSTP Version 1 currently supports only PPP frames. For more information, see [RFC1661].

Name	Value
SSTP_ENCAPSULATED_PROTOCOL_PPP	0x0001

2.2.6 Crypto Binding Request Attribute

The following diagram specifies the format that MUST be used for the Crypto Binding Request attribute. This attribute is used by the SSTP server to communicate the supported hashing methods and nonce to the SSTP client.

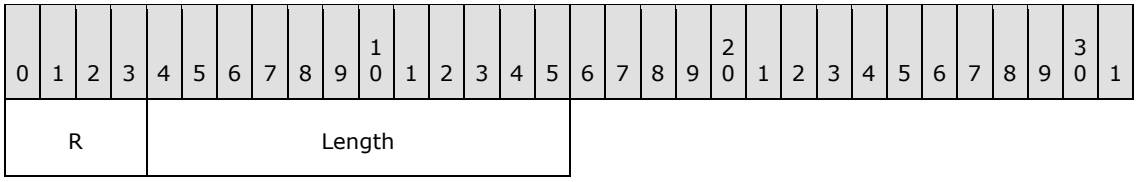
The fields of the structure MUST be transmitted as bytes from left to right.



Reserved (1 byte): This field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x04 for the Crypto Binding Request attribute.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length of the Crypto Binding Request attribute. Its value MUST be 40 bytes (that is, 0x028).

Reserved1 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Hash Protocol Bitmask (1 byte): This 1-byte bitmask field is used to specify the hashing methods that are allowed by the server that is used by the client for computing Compound MAC in the Crypto Binding attribute. For more information, see section 3.1.5.2. The following bits are defined.

Name	Value
CERT_HASH_PROTOCOL_SHA1	0x01
CERT_HASH_PROTOCOL_SHA256	0x02

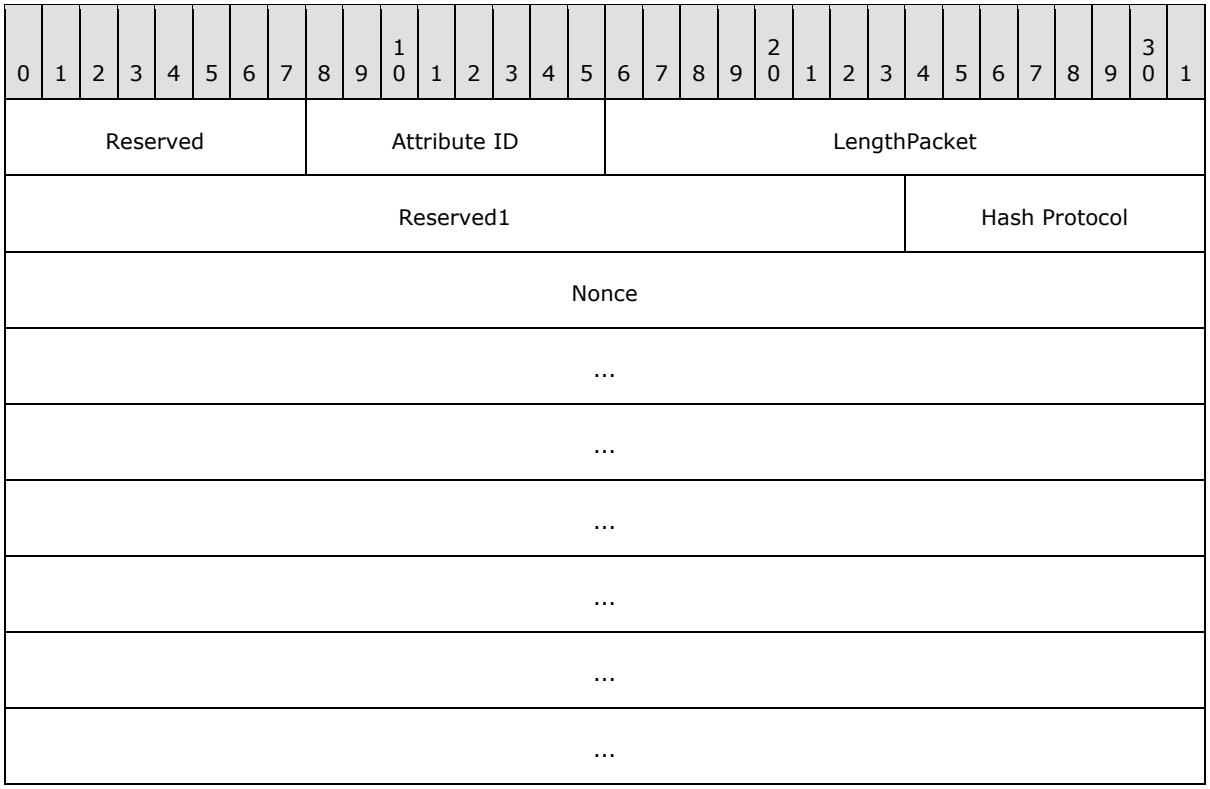
Nonce (32 bytes): A 256-bit unsigned integer that contains a temporally unique (or random) value. For more information, see [RFC1750].

2.2.7 Crypto Binding Attribute

The following diagram specifies the format that MUST be used for the Crypto Binding attribute. This attribute is sent by the SSTP client to the SSTP server and is used to ensure that the SSTP client and SSTP server participated in SSL negotiation and the higher-layer authentication, that is, PPP authentication. For more information, see [RFC1661].

Note that without the Crypto Binding attribute, it is possible for an untrusted man-in-the-middle to relay the PPP packets that are received by the client on another protocol (for example, over wireless) on the SSTP connection without the knowledge of the SSTP client and SSTP server.

The fields of the structure MUST be transmitted as bytes from left to right.

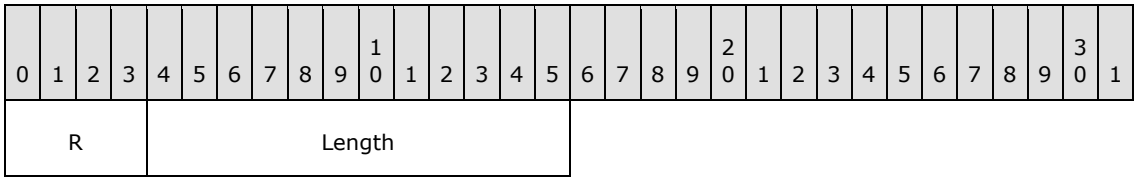


...
Cert Hash (variable)
...
Padding (variable)
...
Compound MAC (variable)
...
Padding1 (variable)
...

Reserved (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x03 for the Crypto Binding attribute.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length of the Crypto Binding attribute. Its value MUST be 104 bytes (that is, 0x068).

Reserved1 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Hash Protocol (1 byte): A 1-byte field that specifies the Cert Hash type and hash algorithm that is used for Compound MAC calculation. Its value MUST be one of the following.

Name	Value
CERT_HASH_PROTOCOL_SHA1	0x01

Name	Value
CERT_HASH_PROTOCOL_SHA256	0x02

Nonce (32 bytes): A 256-bit unsigned integer that contains a temporally unique (or random) value. For more information, see [RFC1750]. This value MUST be the same as what is received from the SSTP server in the Crypto Binding Request attribute.

Cert Hash (variable): A variable-length field in network byte order that contains either the SHA1 hash or the SHA256 hash (as specified by the Cert Hash Type field) of the server certificate that is obtained during SSL/TLS negotiation. This field is either 20 bytes long (when the SHA1 hash is sent) or 32 bytes long (when the SHA256 hash is sent).

Padding (variable): This field is reserved for future use. It MUST be set to 0 and ignored on receipt. This field is either 0 bytes long when the SHA256 Cert Hash is used, or 12 bytes long when the SHA1 Cert Hash is used.

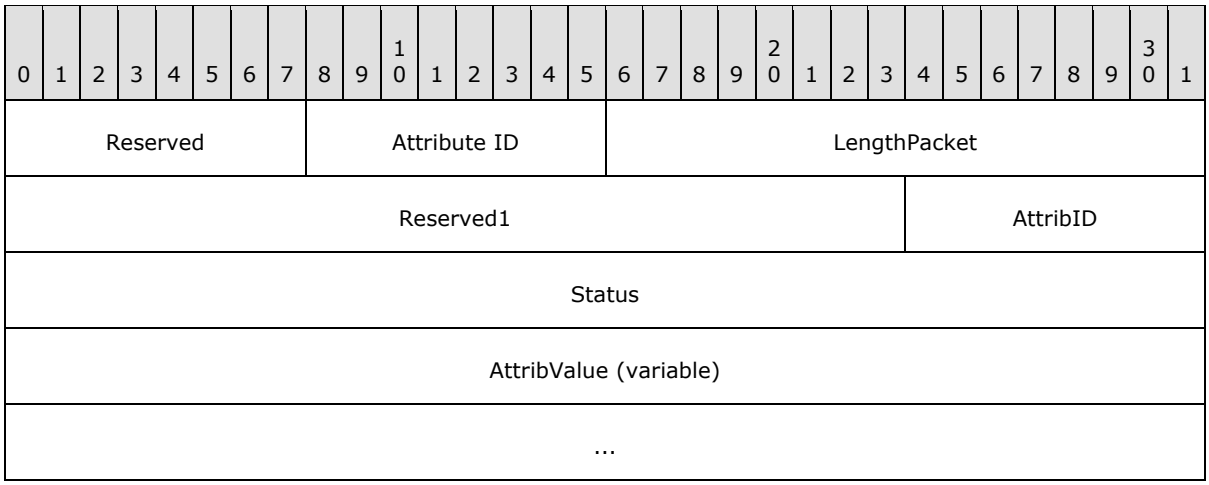
Compound MAC (variable): A variable-length unsigned integer that contains the value that is used to cryptographically associate the higher-layer authentication (that is, PPP authentication) with the lower-layer HTTPS connection; and therefore, ensure that the SSTP client and the SSTP server participated in both of them. (For more information, see section 3.1.5.2 and also see [RFC1661].) This field is either 20 bytes long when the SHA1 Hash Protocol is used for Compound MAC computation, or 32 bytes long when the SHA256 Hash Protocol is used for Compound MAC computation.

Padding1 (variable): A variable-length field that is reserved for future use. It MUST be set to 0 and ignored on receipt. This field is either 0 bytes long when the SHA256 Cert Hash is used, or 12 bytes long when the SHA1 Cert Hash is used.

2.2.8 Status Info Attribute

The following diagram specifies the format that MUST be used for the Status Info attribute. Both the client and server use this attribute to indicate to each other the reason for failure and the unsupported attributes. This attribute can be used to indicate the status of only one attribute at a time; that is, if the server wants to indicate the status of multiple attributes, it should respond with multiple Status Info attributes.

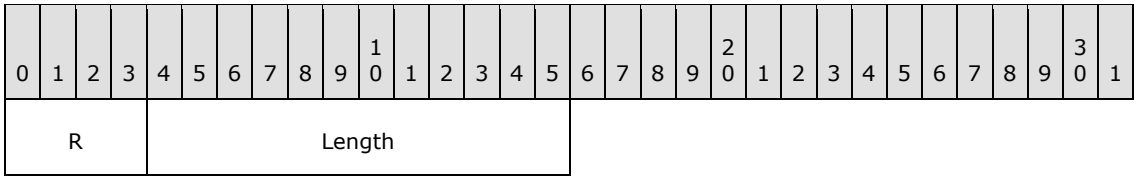
The fields of the structure MUST be transmitted as bytes from left to right.



Reserved (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x2 for the Status Info attribute.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer, in network byte order, that MUST indicate the length of the Status Info attribute. Its value MUST be the length of the AttribValue field plus 12 bytes.

Reserved1 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

AttribID (1 byte): A 1-byte field that specifies the attribute ID whose status is given by the Status Info attribute. Its value MUST be one of the following.

Name	Value
SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID	0x01
SSTP_ATTRIB_STATUS_INFO	0x02
SSTP_ATTRIB_CRYPTO_BINDING	0x03
SSTP_ATTRIB_CRYPTO_BINDING_REQ	0x04

For example, if the Encapsulated Protocol ID that is suggested by the client is not acceptable to the server, it replies with a Status Info attribute that contains AttribID = 0x01 (that is, SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID), Status = 0x00000004 (that is, ATTRIB_STATUS_VALUE_NOT_SUPPORTED), and AttribValue = the value proposed by the client.

Status (4 bytes): A 4-byte field that specifies the reason for the failure. Its value MUST be one of the following.

Value	Meaning
ATTRIB_STATUS_NO_ERROR 0x00000000	This status value MAY be used as a part of the Status Info attribute to indicate a scenario in which there is no error event to report. For example, the higher-layer initiated disconnect is a normal scenario. In such a case, the **SSTP peer** may send a Status Info attribute that has this status value.
ATTRIB_STATUS_DUPLICATE_ATTRIBUTE 0x00000001	This status value MUST be used to indicate multiple occurrences of a specific attribute that is not supported. The AttribID of the Status Info attribute denotes the attribute that is occurring more than one time in the message.
ATTRIB_STATUS_UNRECOGNIZED_ATTRIBUTE 0x00000002	This status value MUST be used to indicate the presence of an unrecognized attribute that is received from the far end. The AttribID field of the Status Info attribute specifies the attribute that is received from the far end that is not recognized by the SSTP peer.
ATTRIB_STATUS_INVALID_ATTRIB_VALUE_LENGTH 0x00000003	This status value MUST be used when the value of the attribute (specified by AttribID in the Status Info attribute) that is received from the **SSTP far end** is of unacceptable length.
ATTRIB_STATUS_VALUE_NOT_SUPPORTED 0x00000004	This status value MUST be used when the value of the attribute (specified by AttribID in the Status Info attribute) that is received from the SSTP far end is not supported by this SSTP peer.
ATTRIB_STATUS_UNACCEPTED_FRAME_RECEIVED 0x00000005	This status value MUST be used when the message type that is received from the far end is not acceptable for the current state of the SSTP peer.
ATTRIB_STATUS_RETRY_COUNT_EXCEEDED 0x00000006	This status value MUST be used when the connection is aborted because the retry count was exceeded for an operation. For example, the SSTP client might fail to provide the acceptable values for the attributes in the Call Connect Request message. If these values are rejected by

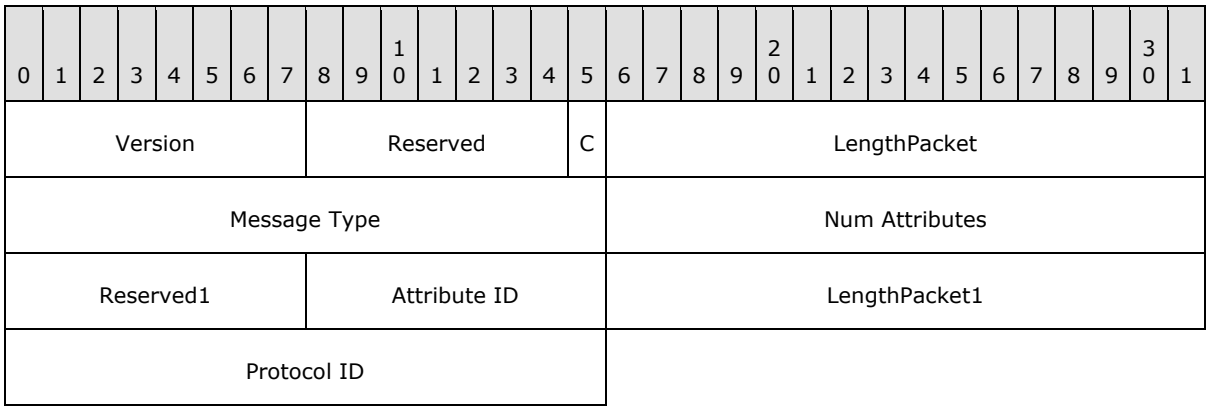
Value	Meaning
	the server with a Call Connect Negative Acknowledgment message for a predefined number of consecutive times,<2> the server terminates the connection and returns this status value.
ATTRIB_STATUS_INVALID_FRAME_RECEIVED 0x00000007	This status MUST be used to stop connections when the message type or the frame format that is received from the SSTP far end is not recognized by the SSTP peer.
ATTRIB_STATUS_NEGOTIATION_TIMEOUT 0x00000008	This status MUST be used to stop connections when the far end has not responded in a timely manner and a time-out occurs.
ATTRIB_STATUS_ATTRIB_NOT_SUPPORTED_IN_MSG 0x00000009	This status MUST be used when sending a negative acknowledgment to a Call Connect Request message when the attribute that is received from the far end is not supported in the specified message. The AttribID in the Status Info attribute indicates the attribute that is not accepted.
ATTRIB_STATUS_REQUIRED_ATTRIBUTE_MISSING 0x0000000a	This status MUST be used in the negative acknowledgment of a Call Connect Request message when a mandatory attribute for the message was not sent by the far end. For example, the SSTP client MUST send SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID as a part of SSTP_MSG_CALL_CONNECT_REQUEST. If this value is not present, the SSTP server MUST send a SSTP_MSG_CALL_CONNECT_NAK that has a status of this value and an AttribID of SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID.
ATTRIB_STATUS_STATUS_INFO_NOT_SUPPORTED_IN_MSG 0x0000000b	This status MUST be used in the SSTP_MSG_CALL_CONNECT_NAK when the client has sent a Status Info attribute that is not supported in the SSTP_MSG_CALL_CONNECT_REQUEST message.

AttribValue (variable): A variable-length field in network byte order that specifies the original value of the attribute that is proposed by the client in the Call Connect Request message that is sent back by the server in the Call Connect Negative Acknowledgment message. It is not present (that is, it is of zero length) in the Call Connect Negative Acknowledgment message, if the mandatory attribute is not sent by the client in the Call Connect Request message; or if the attribute that is sent by the client in the Call Connect Request message is not understood by the server. This field MUST NOT be present (that is, it is of zero length) in any other case where the Status Info attribute is present. The maximum length of this field MUST be 64 bytes. If the original attribute is greater than 64 bytes, only the first 64 bytes are sent back.

2.2.9 Call Connect Request Message

The following diagram specifies the format that MUST be used for the Call Connect Request message. This message MUST be the first message that is sent by the SSTP client after it establishes an HTTPS connection with the server. The client uses this message to request the establishment of an SSTP connection with the server. Note that this message follows the format of an SSTP control packet.

The fields of the structure MUST be transmitted in network byte order from left to right.

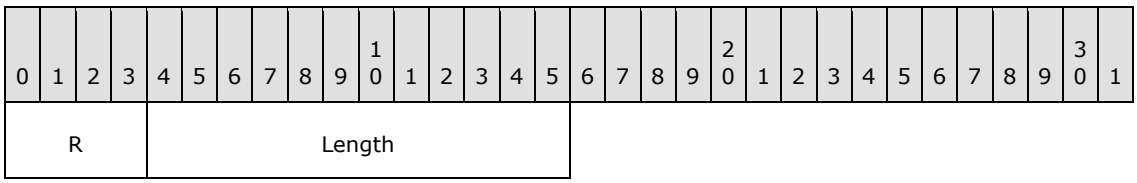


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is used. The upper 4 bits are the MAJOR version, which MUST be 0x1, and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be 1 for a Call Connect Request message that is a control packet.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire Call Connect Request message.

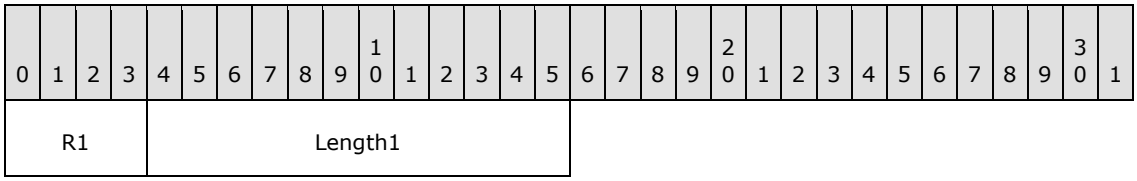
Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. It MUST be 0x0001 (SSTP_MSG_CALL_CONNECT_REQUEST).

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message. This value MUST be 0x0001 because SSTP Version 1 only supports the Encapsulated Protocol ID attribute by using a Protocol ID value of PPP. For more information, see [RFC1661].

Reserved1 (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x01 for the Encapsulated Protocol ID attribute.

LengthPacket1 (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R1 (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

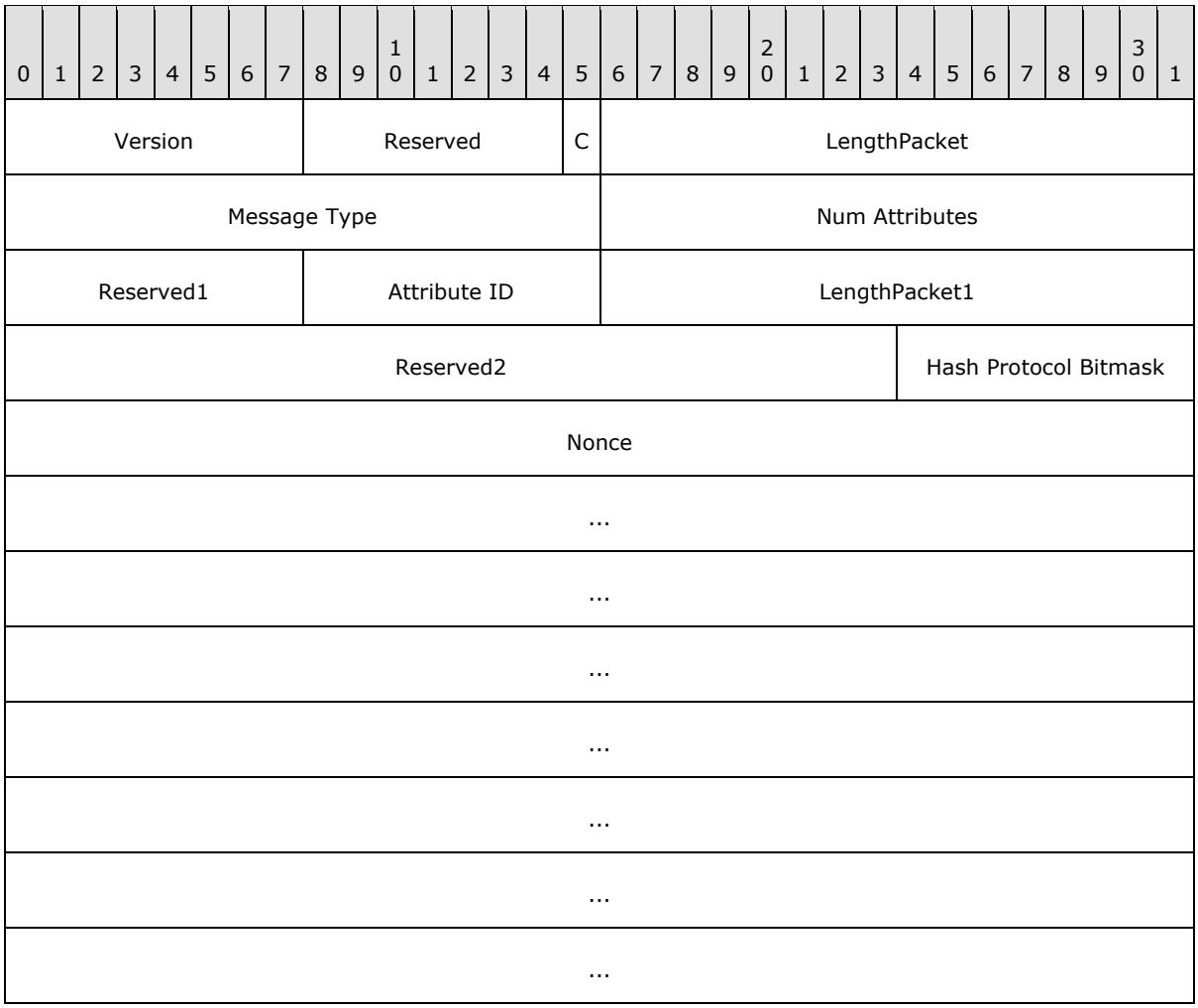
Length1 (12 bits): A 12-bit unsigned integer in network byte order that contains the value 0x006 for the Encapsulated Protocol ID attribute length.

Protocol ID (2 bytes): A 2-byte field in network byte order that contains the value 0x0001 (that is, SSTP_ENCAPSULATED_PROTOCOL_PPP). Note that SSTP Version 1 only supports the transport of PPP frames. For more information, see [RFC1661].

2.2.10 Call Connect Acknowledge Message

The following diagram specifies the format that MUST be used for the Call Connect Acknowledge message. The server sends this message in response to an acceptable Call Connect Request message from a client. Note that this message follows the format of an SSTP control packet (section 2.2.2).

The fields of the structure MUST be transmitted in network byte order from left to right.

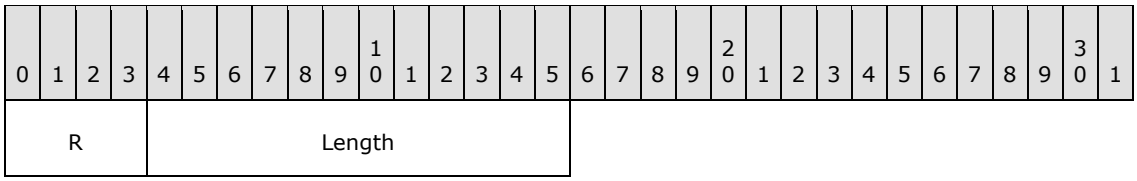


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is being used. The upper 4 bits are the MAJOR version, which MUST be 0x1; and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be 1 for a Call Connect Acknowledge message that is a control packet.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire Call Connect Acknowledge message. This field MUST be set to a value of 48 (0x030).

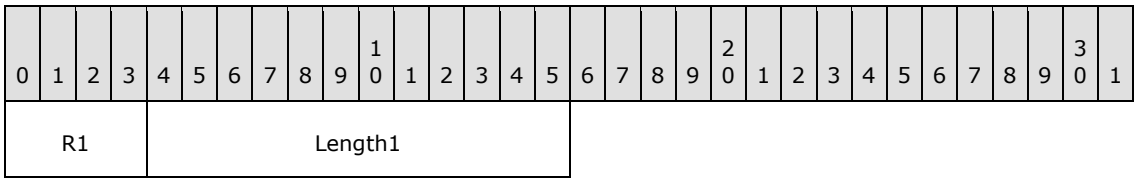
Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. It MUST be 0x0002, that is, SSTP_MSG_CALL_CONNECT_ACK.

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message. This value MUST be 1 for these messages because they contain the Crypto Binding Request attribute only.

Reserved1 (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x04 for the Crypto Binding Request attribute.

LengthPacket1 (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R1 (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length1 (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length of the Crypto Binding Request attribute. Its value MUST be 40 bytes, that is, 0x028.

Reserved2 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Hash Protocol Bitmask (1 byte): This 1-byte bitmask field is used to specify the hashing methods that are allowed by the server for computing Compound MAC in the Crypto Binding attribute. For more information, see section 3.1.5.2. The following bits are defined.

A 1 MUST be placed in the appropriate bit position to select the supported hash protocol. The server MUST select at least one hash protocol. If both are set and the client supports both SHA256 and SHA1 hash protocols, the client MUST select the SHA256 protocol. For more information about how the client processes the **Hash Protocol Bitmask** on receipt of a Call Connect Acknowledge message, see section 3.1.5.3.2.

0	1	2	3	4	5	6	7
0	0	0	0	0	0	B	A

Where the bits are defined as:

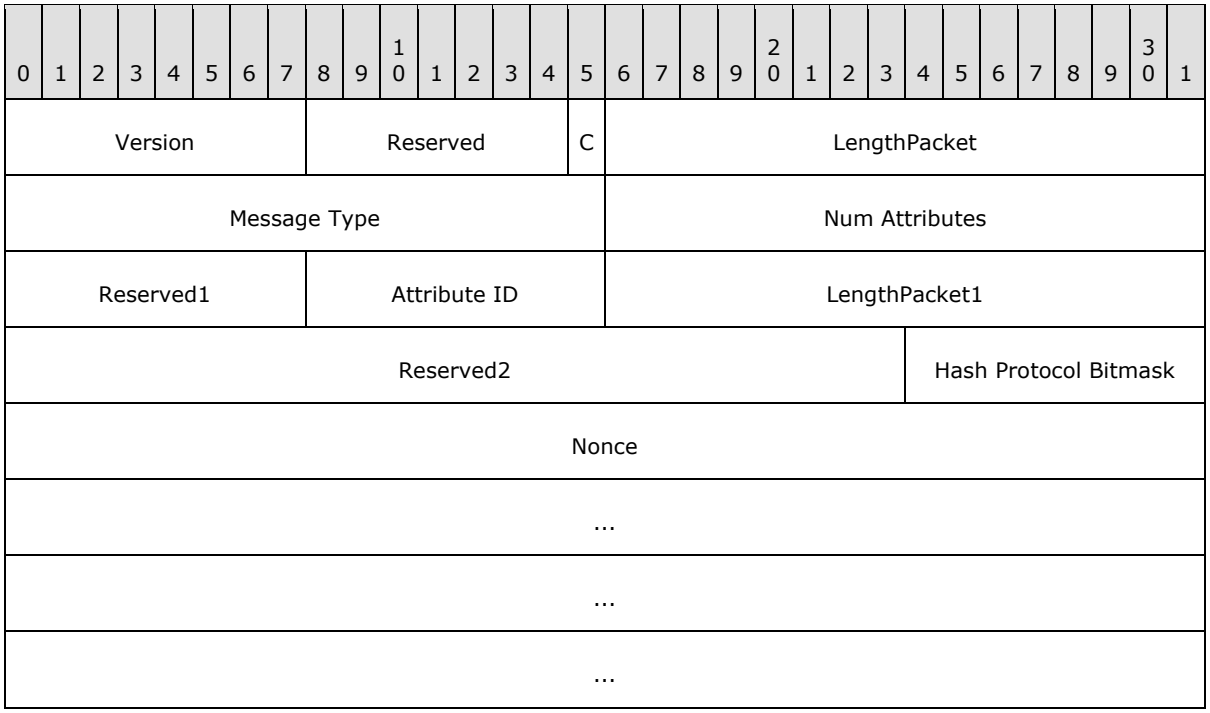
Value	Description
A	CERT_HASH_PROTOCOL_SHA1 is enabled when A=1 and is disabled when A=0.
B	CERT_HASH_PROTOCOL_SHA256 is enabled when B=1 and is disabled when B=0.

Nonce (32 bytes): A 256-bit unsigned integer that contains a temporally unique (or random) value. For more information, see [RFC1750].

2.2.11 Call Connected Message

The following diagram specifies the format that MUST be used for the Call Connected message. This message is sent by the client to the server as a response to the Call Connect Acknowledge message. This message is sent after SSL/TLS negotiation and higher-layer authentication (that is, PPP authentication) is completed. This message marks the completion of SSTP negotiation. It is used to cryptographically bind the SSL/TLS negotiation and PPP authentication so that a man-in-the-middle attacker cannot relay PPP packets that are received on another medium. For example, wireless packets may be received that are not intended for SSTP communication during protocol operation and may represent an attack. For more information about PPP, see [RFC1661].

The fields of the structure MUST be transmitted in network byte order from left to right.



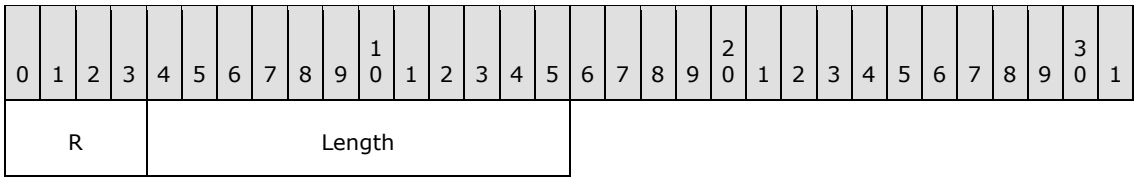
...
...
...
...
Cert Hash (variable)
...
Padding (variable)
...
Compound MAC (variable)
...
Padding1 (variable)
...

Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is being used. The upper 4 bits are the MAJOR version, which MUST be 0x1; and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be 1 for a Call Connect Request message that is a control packet.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length of a Call Connected message. Its value MUST be 112 bytes (that is, 0x070).

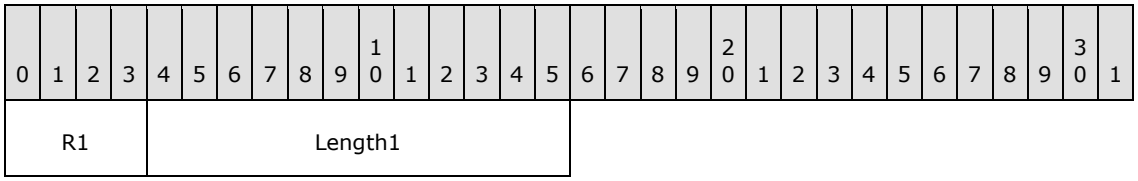
Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. It MUST be 0x0004 (that is, SSTP_MSG_CALL_CONNECTED).

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message. This value MUST be 1 because SSTP only supports the Crypto Binding attribute in a Call Connected message.

Reserved1 (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Attribute ID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute. This value MUST be 0x03 for the Crypto Binding attribute.

LengthPacket1 (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R1 (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length1 (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length of the Crypto Binding attribute. Its value MUST be 104 bytes (that is, 0x068).

Reserved2 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Hash Protocol Bitmask (1 byte): A 1-byte field that specifies the Cert Hash Type and hash algorithm that are used for Compound MAC calculation. Its value MUST be one of the following.

Name	Value
CERT_HASH_PROTOCOL_SHA1	0x01
CERT_HASH_PROTOCOL_SHA256	0x02

Nonce (32 bytes): A 256-bit unsigned integer that contains a temporally unique (or random) value. (For more information, see [RFC1750].) This value MUST be the same as what is

received from the SSTP server in the Call Connect Acknowledge message. This behavior ensures that a man-in-the-middle attacker cannot cause a replay attack.

Cert Hash (variable): A variable-length field in network byte order that contains either the SHA1 hash or the SHA256 hash (as specified by the Cert Hash Type field) of the server certificate that is obtained during SSL negotiation. The length of the field is either 20 bytes long when the SHA1 hash is used or 32 bytes long when the SHA256 hash is used.

Padding (variable): A variable-length field reserved for future use that MUST be set to 0 by the sender and ignored by the receiver. This field is either 0 bytes long when the SHA256 Cert Hash is used or 12 bytes long when the SHA1 Cert Hash is used.

Compound MAC (variable): A variable-length unsigned integer that contains the value that is used to cryptographically associate the higher-layer authentication (that is, PPP authentication) with a lower-layer HTTPS connection; and therefore, ensure that the SSTP client and the SSTP server participated in both PPP authentication and HTTPS connection. (For more information, see section 3.1.5.2 and also see [RFC1661].) This field is either 20 bytes long when the SHA1 Hash Protocol is used for Compound MAC computation or 32 bytes long when the SHA256 Hash Protocol is used for Compound MAC computation.

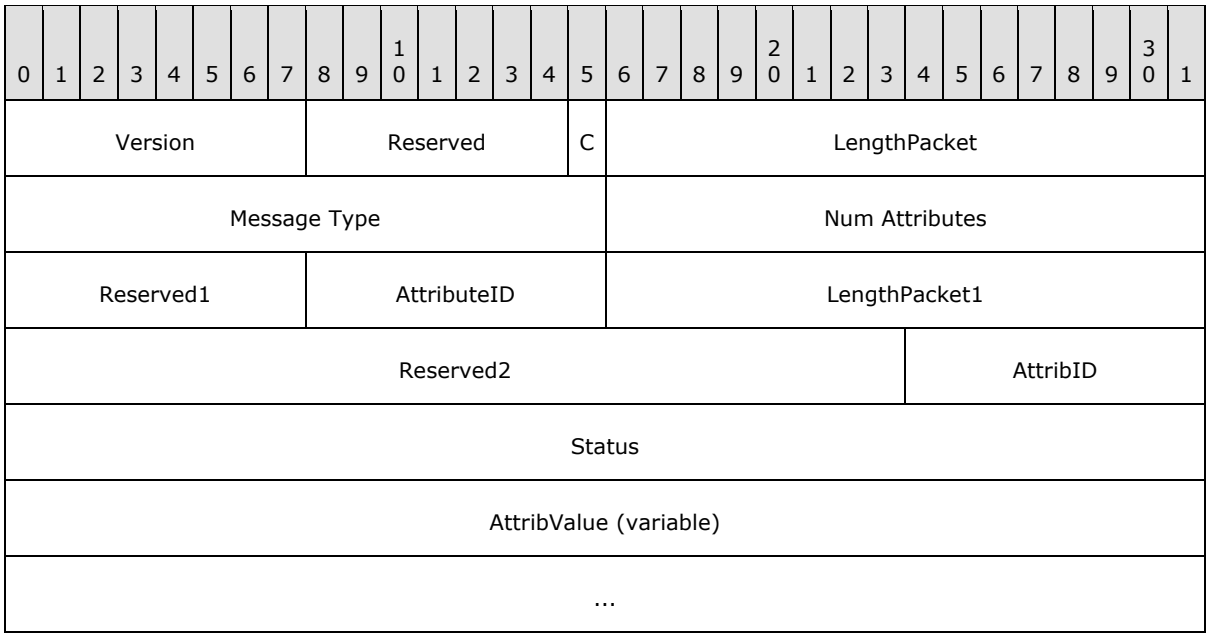
Padding1 (variable): A variable-length field that is reserved for future use that MUST be set to 0 by the sender and ignored by the receiver. This field is either 0 bytes in length when the SHA256 Hash Protocol is used for Compound MAC computation, or 12 bytes in length when the SHA1 Hash Protocol is used.

Value	Meaning
0	Reserved

2.2.12 Call Connect Negative Acknowledgment Message

The following diagram specifies the format that MUST be used for the Call Connect Negative Acknowledgment message. This message is sent by the SSTP server in response to an unacceptable Call Connect Request message that is sent by the SSTP client. Note that this message follows the format of an SSTP control packet, as specified in 2.2.2. This message MUST have one or more Status Info attributes. This message MUST NOT have any other attribute.

The fields of the structure MUST be transmitted in network byte order from left to right.

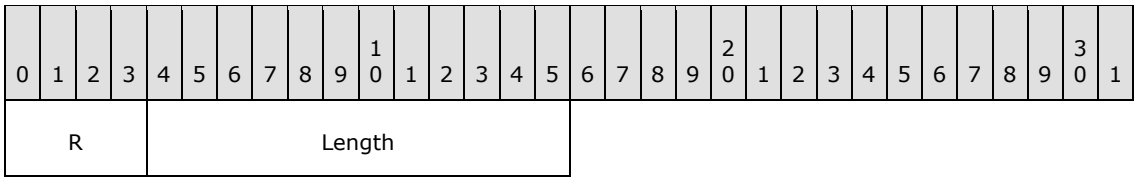


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is being used. The upper 4 bits are MAJOR version, which MUST be 0x1 and the lower 4 bits are MINOR version, which MUST be set to 0x0. Therefore, the 8-bit value of the Version field MUST be 0x10 and corresponds to Version 1.0.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or an SSTP data packet. The value MUST be 1 for a Call Connect Negative Acknowledgment message that is a control packet.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order that MUST indicate the length, in bytes, of the entire Call Connect Negative Acknowledgment message.

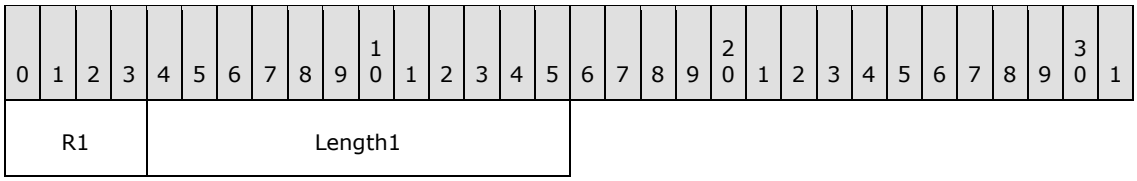
Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. It MUST be 0x0003 (that is, SSTP_MSG_CALL_CONNECT_NAK).

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message. This value MUST be greater than or equal to 1 because this message MUST have at least one Status Info attribute.

Reserved1 (1 byte): This 8-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

AttributeID (1 byte): An 8-bit (1-byte) field that is used to specify the type of attribute; its value MUST be 0x02 for the Status Info attribute.

LengthPacket1 (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R1 (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length1 (12 bits): A 12-bit unsigned integer in network byte order that contains the value 12 plus the length of the AttribValue field.

Reserved2 (3 bytes): This 24-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

AttribID (1 byte): A 1-byte field that specifies the attribute ID whose status is given by the Status Info attribute. Its value MUST be one of the following.

Name	Value
SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID	0x01
SSTP_ATTRIB_STATUS_INFO	0x02
SSTP_ATTRIB_CRYPTO_BINDING	0x03
SSTP_ATTRIB_CRYPTO_BINDING_REQ	0x04

For example, if the Encapsulated Protocol ID that is suggested by the client is not acceptable to the server, it replies with a Status Info attribute that contains AttribID = 0x01 (that is, SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID), Status = 0x00000004 (that is, ATTRIB_STATUS_VALUE_NOT_SUPPORTED), and AttribValue = value proposed by the client.

Status (4 bytes): A 4-byte field that specifies the reason for the failure. Its value MUST be one of the following values, the description of which, is specified in the Status Info attribute.

Value	Meaning
ATTRIB_STATUS_NO_ERROR 0x00000000	This Status value MAY be used as part of Status Info to indicate the scenario in which there is no error event to report. For example, the higher layer initiated disconnect will be a normal scenario. In such a case, the SSTP peer may send a Status Info attribute with this status value.
ATTRIB_STATUS_DUPLICATE_ATTRIBUTE	This status value MUST be used to indicate

Value	Meaning
0x00000001	multiple occurrences of a specific attribute that is not supported. The AttribID of the Status Info attribute denotes the attribute that is occurring more than one time in the message.
ATTRIB_STATUS_UNRECOGNIZED_ATTRIBUTE 0x00000002	This status value MUST be used to indicate the presence of an unrecognized attribute that is received from the far end. The AttribID field of the Status Info attribute specifies the attribute that is received from the far end that is not recognized by the SSTP peer.
ATTRIB_STATUS_INVALID_ATTRIB_VALUE_LENGTH 0x00000003	This status value MUST be used when the value of the attribute (specified by AttribID in the Status Info attribute) that is received from the SSTP far end is of unacceptable length.
ATTRIB_STATUS_VALUE_NOT_SUPPORTED 0x00000004	This status value MUST be used when the value of the attribute (specified by AttribID in the Status Info attribute) that is received from the SSTP far end is not supported by this SSTP peer.
ATTRIB_STATUS_UNACCEPTED_FRAME_RECEIVED 0x00000005	This status value MUST be used when the message type that is received from the far end is not acceptable for the current state of the SSTP peer.
ATTRIB_STATUS_RETRY_COUNT_EXCEEDED 0x00000006	This status value MUST be used when the connection is being aborted because the retry count is exceeded for an operation. For example, if the SSTP client failed to provide the acceptable values for the attributes in the connect request that are rejected by the server with a call connect NAK for a predefined number <8> of consecutive times, the server aborts the connection and uses this status value.
ATTRIB_STATUS_INVALID_FRAME_RECEIVED 0x00000007	This status MUST be used to abort connections when the message type or the frame format that is received from the SSTP far end is not recognized by the SSTP peer.
ATTRIB_STATUS_NEGOTIATION_TIMEOUT 0x00000008	This status MUST be used to abort connections when the far end has not responded in a timely manner and a time-out results.
ATTRIB_STATUS_ATTRIB_NOT_SUPPORTED_IN_MSG 0x00000009	This status MUST be used when sending a negative acknowledgment to a call connect request when the attribute that is received from the far end is not supported in the specified message. The AttribID in the Status Info attribute indicates the attribute

Value	Meaning
	that is not accepted.
ATTRIB_STATUS_REQUIRED_ATTRIBUTE_MISSING 0x0000000a	This status MUST be used in the negative acknowledgment of a call connect request when a mandatory attribute for the message has not been sent by the far end. For example, the SSTP client MUST send SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID as part of an SSTP_MSG_CALL_CONNECT_REQUEST. If this is not present, the SSTP server MUST send SSTP_MSG_CALL_CONNECT_NAK with the status being this value and an AttribID of SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID.
ATTRIB_STATUS_STATUS_INFO_NOT_SUPPORTED_IN_MSG 0x0000000b	This status MUST be used in SSTP_MSG_CALL_CONNECT_NAK when the client has sent a Status Info attribute that is not supported in the SSTP_MSG_CALL_CONNECT_REQUEST message.

AttribValue (variable): A variable-length field, in network byte order, that specifies the original value of the attribute that is proposed by the client in the Call Connect Request message that is sent back by the server in the Call Connect Negative Acknowledgment message. This field is not present (that is, it is of zero length) in the Call Connect Negative Acknowledgment message if the mandatory attribute is not sent by the client in the Call Connect Request message; or if the attribute that is sent by the client in the Call Connect Request message is not understood by the server. This field MUST NOT be present (that is, it is of zero length) in any other case where the Status Info attribute is present. The maximum length of this field may be 64 bytes. If the original attribute is greater than 64 bytes, only the first 64 bytes are sent back.

2.2.13 Call Abort and Call Disconnect Messages

The Call Abort message is sent by an SSTP peer to the SSTP far end in order to initiate an abnormal disconnection of the SSTP connection. This behavior occurs when an invalid message is received as specified in the state transition diagrams that are specified in sections 3.3.1.1.1, 3.1.1.1.1, and 3.2.1.1.1. For example, if the client does not send the Call Connect Request message as the first SSTP message, the SSTP server MUST send a Call Abort message to the SSTP client to tear down the SSTP connection.

The Call Disconnect message is sent by both the SSTP client and the SSTP server to the SSTP far end in order to initiate a normal disconnection of the SSTP connection. For example, a user initiates a manual disconnect.

Both the Call Abort message and the Call Disconnect message are similar to the Call Connect Negative Acknowledgment message (that is, one or more Status Info attributes are present). However, unlike the Call Connect Negative Acknowledgment message in which there MUST be a Status Info attribute present, the Call Abort message and the Call Disconnect message do not mandate the presence of a Status Info attribute.<3>

2.2.14 Call Disconnect Acknowledge, Echo Request, and Echo Response Messages

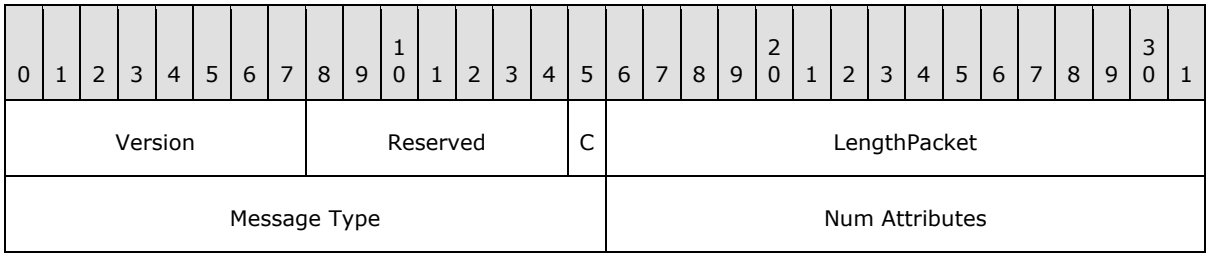
The Call Disconnect Acknowledge, Echo Request, and Echo Response messages are identical in wire format except for the message type field.

A Call Disconnect Acknowledge message can be sent by both the SSTP client and the SSTP server to each other (that is, to the SSTP far end) in response to a Call Disconnect message that is received from the SSTP far end.

The Echo Request message is sent by both the SSTP client and the SSTP server when they do not receive an SSTP packet (either an SSTP control packet or SSTP data packet) in a specified amount of time.

The Echo Response message is sent by both the SSTP client and the SSTP server to each other in response to an Echo Request packet that is received from one another.

The fields of the structure MUST be transmitted in network byte order from left to right.

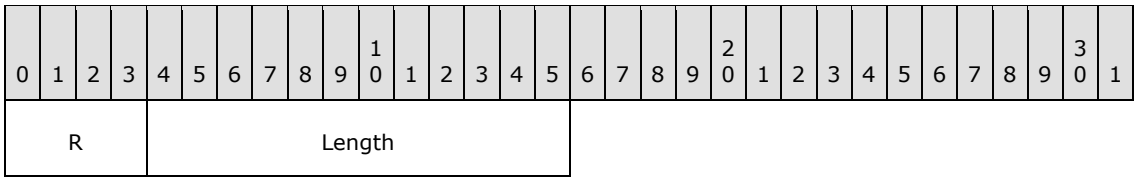


Version (1 byte): An 8-bit (1-byte) field that is used to communicate and negotiate the version of SSTP that is used. The upper 4 bits are the MAJOR version, which MUST be 0x1, and the lower 4 bits are the MINOR version, which MUST be set to 0x0. This means that the 8-bit value of the Version field MUST be 0x10.

Reserved (7 bits): This 7-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

C (1 bit): A 1-bit field that is used to indicate whether the packet is an SSTP control packet or SSTP data packet. The value MUST be set to 1.

LengthPacket (2 bytes): A 16-bit unsigned integer in network byte order that packs data for two fields, configured in the following format.



R (4 bits): This 4-bit field is reserved for future use. It MUST be set to 0 and ignored on receipt.

Length (12 bits): A 12-bit unsigned integer in network byte order. This field indicates the length, in bytes, of the entire message. This field MUST be set to 8 (0x008).

Message Type (2 bytes): A 16-bit field in network byte order that specifies the type of message. This field MUST be one of the following values.

Name	Value
SSTP_MSG_CALL_DISCONNECT_ACK	0x0007
SSTP_MSG_ECHO_REQUEST	0x0008
SSTP_MSG_ECHO_RESPONSE	0x0009

Num Attributes (2 bytes): A 16-bit field in network byte order that specifies the number of attributes in the message. This value MUST be 0 for these messages because they do not support any attribute.

3 Protocol Details

The following sections specify details of the Secure Socket Tunneling Protocol (SSTP), including client, server, and common abstract data models; and message processing rules.

3.1 Client Details

The following section gives details of the SSTP client.

3.1.1 Abstract Data Model

See section 3.3.1.

3.1.1.1 State Machine

3.1.1.1.1 Call Establishment

The following figure shows the **state machine** when a client establishes the outgoing SSTP tunnel.

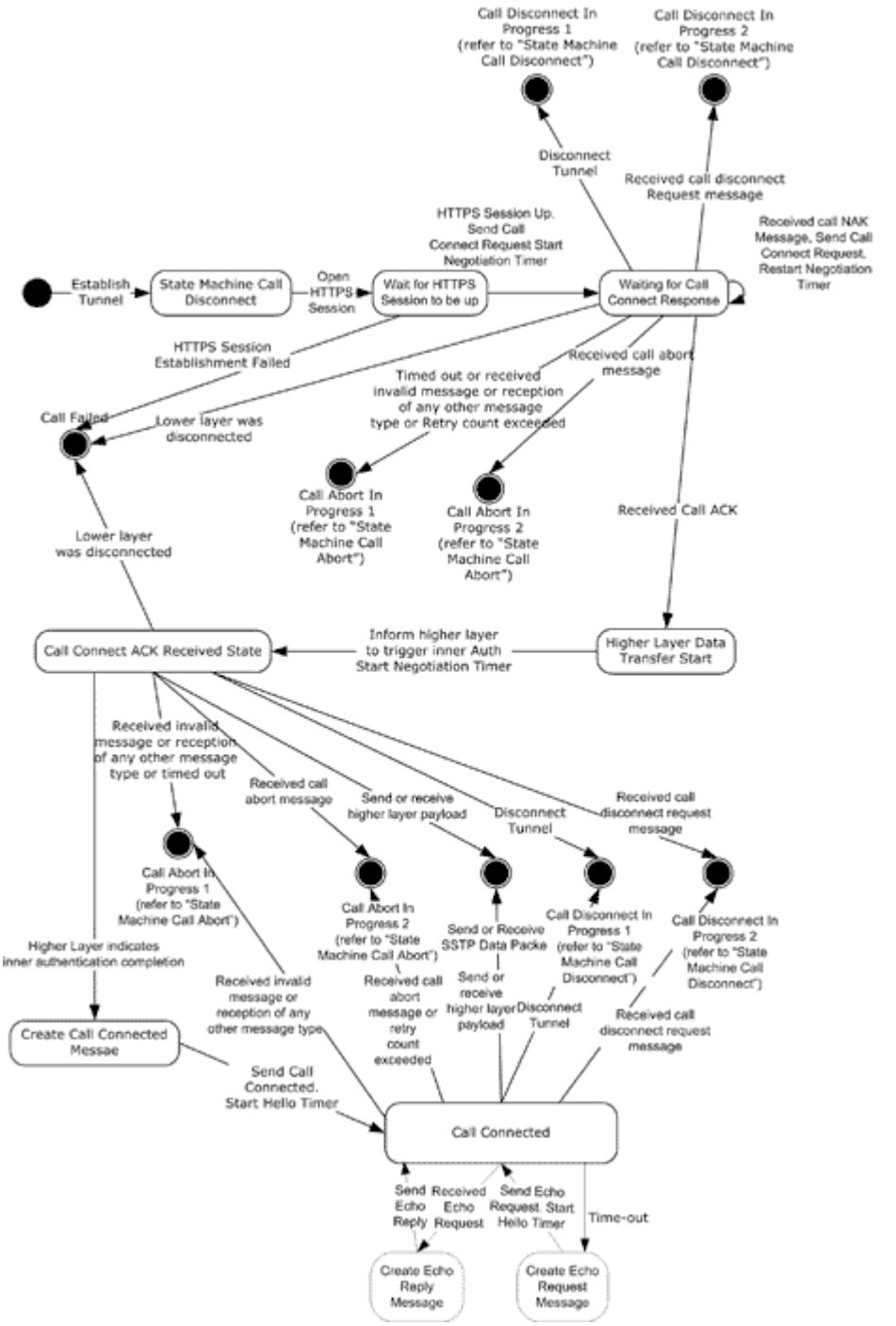


Figure 2: Client call establishment

3.1.2 Timers

For common timers, see section 3.3.2.

3.1.2.1 Negotiation Timer

When establishing an SSTP connection, the SSTP client starts the Negotiation timer after sending a Call Connect Request message and after receiving a Call Connect Acknowledge message.

The client MAY<4> choose to implement different values of timer after sending the Call Connect Request message and after receiving the Call Connect Acknowledge message. The timer SHOULD be set to 60 seconds.

3.1.3 Initialization

Client side initialization is performed when the user tries to establish an SSTP tunnel to the SSTP server. This process is specified in section 3.1.4.1.

3.1.4 Higher-Layer Triggered Events

The primary trigger events for SSTP are to establish the tunnel, cancel a tunnel under progress, and disconnect an established tunnel.

3.1.4.1 Establish Tunnel Event

When the client establishes a tunnel to the remote SSTP server, it initiates the tunnel request to the SSTP layer. The SSTP layer MUST first establish a bidirectional HTTPS session, for example, see section 4.1. The SSTP client MUST use content length encoding together with the following parameters:<5>

```
Method: SSTP_DUPLEX_POST
Protocol Version: HTTP/1.1
```

After the HTTPS layer is set up, the SSTP state machine MUST be initiated. The client, which MUST be configured for PPP over SSTP, MUST send a Call Connect Request message with the Encapsulated Protocol ID that corresponds to PPP. For more information about PPP, see [RFC1661].

3.1.4.2 Disconnect Tunnel Event

When the SSTP tunnel is no longer required, it MUST be brought down by using the Disconnect Tunnel request. The disconnect tunnel event MAY be used to bring down the established tunnel or a call setup that is in progress.

For an established tunnel, this request typically comes after the encapsulated protocol has completed its teardown (in this specific implementation, it is the PPP). After the PPP is torn down, SSTP is triggered to bring down the tunnel. The SSTP peer initiates the disconnection by sending the Call Disconnect message to the SSTP far end. For more information about PPP, see [RFC1661].

For a call setup in progress, the disconnect event MAY occur due to cancellation of the Establish Tunnel request. In this case, irrespective of the current state, the SSTP peer MUST send a Call Disconnect message to the SSTP far end. The SSTP far end MUST acknowledge the disconnect request irrespective of the current SSTP state.

After the SSTP message exchange is completed, the HTTPS layer MUST be notified about this event, and the HTTPS layer SHOULD eventually be brought down.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Status and Error Handling

See section 3.3.5.1.

3.1.5.2 Crypto Binding

During the SSL/TLS negotiation, as part of establishing an HTTPS connection, the SSTP client authenticates the SSTP server. However, it is optional for the SSTP server to authenticate the client. The client is authenticated by the server during the higher-layer authentication (that is, PPP authentication). Therefore, it is possible for a man in the middle to establish the HTTPS connection to the SSTP server and forward the PPP packets that it received from a client for a communication other than SSTP communications (for example, wireless communications). To prevent such attacks, it is important to cryptographically bind the two authentications.

SSTP implements cryptographic binding by requiring the client to send a value over the HTTPS connection as an SSTP message. This value is derived from the keying material that is generated during PPP authentication. By using this value, the SSTP client can prove that it is the entity that was authenticated with the SSTP server, and that the PPP authentication was used for SSTP communications.

Because the client already authenticated the SSTP server during SSL/TLS negotiation as part of establishing an HTTPS connection, the client can also confirm from the SSTP server that there is no man in the middle or that the entity between the client and server is an entity the SSTP server trusts (see section 5). This process, which is termed crypto binding, is used to protect the SSTP negotiation against man-in-the-middle attacks.

The sequence of steps that occurs is as follows:

- The TCP connection is established by the SSTP client to the SSTP server over TCP port 443.
- SSL/TLS negotiation is completed over this TCP connection. The SSTP server is authenticated by the SSTP client. However, the client authentication by the server is only optional.
- The HTTPS request-response is completed.
- SSTP negotiation begins. The SSTP client sends a Call Connect Request message to the SSTP server. The SSTP server validates the request and sends a Call Connect Acknowledge message that contains a nonce to be used by the SSTP client in the Call Connected message.
- PPP negotiation is initiated, and PPP authentication is completed. For more information about PPP, see [RFC1661].
- The SSTP client sends the Call Connected message (section 2.2.11) that contains the nonce that was sent by the SSTP server, the hash of the server certificate that is used in SSL/TLS negotiation, and the Compound MAC that is computed by using the keying material that is generated in PPP authentication. The SSTP server validates the Call Connected message, and SSTP negotiation is completed.
- PPP negotiation (that is, a network control protocol such as IP Control Protocol (IPCP) is negotiated) is completed.

Implementations MUST support the Crypto Binding feature of SSTP.

SSTP Version 1 allows either SHA1 or SHA256 to be used for generating the Compound MAC field in the Call Connected message.

If SHA1 is used, the Compound MAC field in the Crypto Binding packet MUST contain the output of an HMAC-SHA1-160 operation (as specified in [RFC2104] and [RFC3174]), in which the key is derived from the higher-layer authentication method (that is, the PPP authentication method in SSTP Version 1). For information about how an implementation generates the data that is used in the HMAC-SHA1-160 operation for the Crypto Binding packet, see section 3.1.5.2.1. For information about how an implementation generates the key that is used in the HMAC-SHA1-160 operation for the Crypto Binding packet, see section 3.1.5.2.2.

If SHA256 is used, the Compound MAC field in the Crypto Binding packet MUST contain the output of an HMAC-SHA256-256 operation (as specified in [SHA256]), in which the key is derived from the higher-layer authentication method (that is, the PPP authentication method in SSTP Version 1). For information about how an implementation generates the data that is used in the HMAC-SHA256-256 operation for the Crypto Binding packet, see 3.1.5.2.3. For information about how an implementation generates the key that is used in the HMAC-SHA256-256 operation for the Crypto Binding packet, see section 3.1.5.2.4.

For information about how the semantics are used by the SSTP client and server when performing the Crypto Binding exchanges, see sections 3.1.5.3.2 and 3.2.5.2.3.

3.1.5.2.1 Input Data Used in the Crypto Binding HMAC-SHA1-160 Operation

The data that is used as the input to the HMAC-SHA1-160 operation and also used in the creation of the Compound MAC MUST be constructed from the entire 112 bytes of the Call Connected message (section 2.2.11). In this case, the Compound MAC field and Padding field MUST be zeroed out.

3.1.5.2.2 Key Used in the Crypto Binding HMAC-SHA1-160 Operation

The key that is used as the input to the HMAC-SHA1-160 operation and used in the creation of the Compound MAC MUST be constructed by following the steps that are specified in the following sections. These steps produce the following intermediate values, which are defined later in this section:

- **Higher-Layer Authentication Key (HLAK)**

  First, a 32-byte long string is generated from keys that are provided by the higher-layer PPP authentication method.

  If the higher-layer PPP authentication method generates Microsoft Point-to-Point Encryption (MPPE) keys, as specified in [RFC3079], an implementation MUST obtain the HLAK by using the following method:

  - For MS-CHAPv2, as specified in [RFC2759]:

    SSTP Client HLAK = MasterSendKey | MasterReceiveKey, and:

    SSTP Server HLAK = MasterReceiveKey | MasterSendKey,

    where | indicates concatenation of strings, and MasterSendKey and MasterReceiveKey are as specified in [RFC3079] section 3.

  - For EAP TLS, as specified in [RFC2716]:

    SSTP Client HLAK = MasterSendKey | MasterReceiveKey, and:

    SSTP Server HLAK = MasterReceiveKey | MasterSendKey,

where | indicates concatenation of strings and MasterSendKey and MasterReceiveKey are as specified in [RFC3079] section 4.

- For EAP (other than EAP TLS), as specified in [RFC2284]:

  SSTP Client HLAK = Client Master Session Key (MSK), as specified in [RFC3748], and:

  SSTP Server HLAK = Server Master Session Key (MSK), as specified in [RFC3748].

If the HLAK is more than 32 octets, the first 32 octets form the HLAK. If the HLAK is less than 32 octets, the string is padded with 0x00 at the end to obtain a total length of 32 octets.

If the higher-layer PPP authentication method did not generate any keys, the HLAK MUST be 32 octets of 0x00.

- **Compound MAC Key Seed**

  Next, the seed value is generated. An implementation MUST create a byte array of size 29 bytes containing the ASCII values for the string "SSTP inner method derived CMK" which will be used as Compound MAC Key Seed value.

- **Compound MAC Key (CMK)**

  Finally, the PRF+ operation generates the key to be used to derive the Compound MAC using the HMAC-SHA1-160 operation.

  To generate the Compound MAC Key (CMK), implementations MUST use the HLAK, MUST use the PRF+ seed value as the input to a PRF+ operation, and MUST generate 32 bytes:

  CMK = First 20 octets of PRF+ (HLAK, CMK Seed, 20);

  The PRF algorithm is based on PRF+ from IKEv2 (for more information, see [RFC4306] section 2.13) shown below ("|" denotes concatenation):

  - K = Key, S = Seed, LEN = output length, represented as binary in a single octet.
  - PRF (K, S, LEN) = T1 | T2 | T3 | T4 | ... where:
    - T1 = HMAC-SHA1 (K, S | LEN | 0x01)
    - T2 = HMAC-SHA1 (K, T1 | S | LEN | 0x02)
    - T3 = HMAC-SHA1 (K, T2 | S | LEN | 0x03)
    - T4 = HMAC-SHA1 (K, T3 | S | LEN | 0x04)
    - ...

3.1.5.2.3 Input Data Used in the Crypto Binding HMAC-SHA256-256 Operation

The data that is used as the input to the HMAC-SHA256-256 operation and used in the creation of the Compound MAC MUST be constructed from the entire 112 bytes of the Call Connected message (section 2.2.11) with the Compound MAC field and Padding field zeroed out.

3.1.5.2.4 Key Used in the Crypto Binding HMAC-SHA256-256 Operation

The key that is used as the input to the HMAC-SHA256-256 operation and used in the creation of the Compound MAC MUST be constructed by following the steps that are specified in the following

sections. These steps produce the following intermediate values that are defined later in this section:

- **Higher-Layer Authentication Key (HLAK)**

  First, a 32-byte long string is generated from keys that are provided by the higher-layer PPP authentication method.

  If the higher-layer PPP authentication method generates Microsoft Point-to-Point Encryption (MPPE) keys, as specified in [RFC3079], an implementation MUST obtain the HLAK using the following method:

  - For MS-CHAPv2, as specified in [RFC2759]:

    SSTP Client HLAK = MasterSendKey | MasterReceiveKey and:

    SSTP Server HLAK = MasterReceiveKey | MasterSendKey,

    where | indicates concatenation of strings and MasterSendKey and MasterReceiveKey are as specified in [RFC3079] section 3.

  - For EAP TLS, as specified in [RFC2716]:

    SSTP Client HLAK = MasterSendKey | MasterReceiveKey and:

    SSTP Server HLAK = MasterReceiveKey | MasterSendKey,

    where MasterSendKey and MasterReceiveKey are as specified in [RFC3079] section 4 and where | indicates concatenation of strings.

  - For EAP, other than EAP TLS, as specified in [RFC2284]:

    SSTP Client HLAK = Client Master Session Key (MSK), as specified in [RFC3748] and:

    SSTP Server HLAK = Server Master Session Key (MSK), as specified in [RFC3748].

  If the HLAK is more than 32 octets, the first 32 octets form the HLAK. If the HLAK is less than 32 octets, the string is padded with 0x00 at the end to obtain a total length of 32 octets.

  If the higher-layer PPP authentication method did not generate any keys, the HLAK MUST be 32 octets of 0x00.

- **Compound MAC Key Seed**

  Next, the seed value is generated. An implementation MUST create a byte array of size 29 bytes containing the ASCII values for the string "SSTP inner method derived CMK" which will be used as Compound MAC Key Seed value.

- **Compound MAC Key (CMK)**

  Finally, the PRF+ operation generates the key to be used to derive the Compound MAC using the HMAC-SHA256-256 operation.

  To generate the Compound MAC Key (CMK), implementations MUST use the HLAK, MUST use the PRF+ seed value as the input to a PRF+ operation, and MUST generate 32 bytes.

  CMK = First 32 octets of PRF+ (HLAK, CMK Seed, 32);

The PRF algorithm is based on PRF+ from IKEv2 (for more information, see [RFC4306] section 2.13) shown below ("|" denotes concatenation):

- K = Key, S = Seed, LEN = output length, represented as binary in a single octet.
- PRF (K, S, LEN) = T1 | T2 | T3 | T4 | ... where:
  - T1 = HMAC-SHA256 (K, S | LEN | 0x01)
  - T2 = HMAC-SHA256 (K, T1 | S | LEN | 0x02)
  - T3 = HMAC-SHA256 (K, T2 | S | LEN | 0x03)
  - T4 = HMAC-SHA256 (K, T3 | S | LEN | 0x04)
  - ...

3.1.5.3 Packet Processing

3.1.5.3.1 General Packet Validation

When receiving a packet, the SSTP peer MUST validate that the packet conforms to the syntax as specified in Message Syntax and its subsections. If an invalid packet is received, see section 3.3.5.1 for error handling.

3.1.5.3.2 Receiving a Call Connect Acknowledge Message

The Call Connect Acknowledge message MUST be received by an SSTP client in response to the Call Connect Request packet that is sent by the SSTP client before the negotiation timer expires. SSTP Version 1 only supports PPP as the higher-layer protocol. The client MUST verify that the Crypto Binding Request attribute is present, at least one hash protocol is selected by the server in the **Hash Protocol Bitmask** field, and at least one of the hash protocols in the **Hash Protocol Bitmask** field is supported by it. If both the server and the client support SHA256 and SHA1, the client MUST use the SHA256 hash protocol in the Call Connected message. For error handling, see section 3.3.5.1.

If the packet is acceptable, the client MUST trigger PPP to start the PPP state machine. After PPP authentication is completed, it MUST respond with a Call Connected message that contains the same nonce value that the server sent in the Call Connect Acknowledge message. The message MUST contain the Crypto Binding attribute. For how to construct the Crypto Binding attribute, see section 3.1.5.2. Also, the client triggers the PPP state machine after sending the Call Connected message.

3.1.5.3.3 Receiving a Call Connect Negative Acknowledgment Message

The Call Connect Negative Acknowledgment message MUST be received by an SSTP client only in response to the Call Connect Request packet that is sent by the SSTP client before the negotiation timer expires. Otherwise, the SSTP peer MUST start abort processing of the connection by sending a Call Abort message. (Starting in SSTP Version 1, only the Encapsulation Protocol ID is sent in the Call Connect Request message, and a negative acknowledge message should not be received by the client. If the client receives a negative acknowledge message, this event indicates an error in the SSTP client or server implementation.) The first time this message is received by the client in response to a particular Call Connect Request packet, the client SHOULD retry an implementation-dependent number of times<6> by sending a Call Connect Request message.

3.1.5.3.4 Receiving a Call Abort Message

If the Call Abort message is received in response to the Call Abort message that is sent by the SSTP client, the client MUST clean up the connection immediately. If the SSTP client did not send a Call Abort message, the client MUST respond with a Call Abort message, wait for a time of length TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.1), and then clean up the connection.

3.1.5.3.5 Receiving a Call Disconnect Message

If the SSTP client receives a Call Disconnect message, the client MUST respond with a Call Disconnect Acknowledge message, wait for a time of length TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.2), and then clean up the connection.

3.1.5.3.6 Receiving a Call Disconnect Acknowledge Message

If the Call Disconnect Acknowledge message is received in response to the Call Disconnect message that is sent by the SSTP client, the client MUST clean up the connection immediately. If the SSTP client did not send a Call Disconnect message, the client MUST respond with a Call Abort message, wait for a time of length TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.1), and then clean up the connection.

3.1.5.3.7 Receiving an Echo Request Message

If the SSTP client receives an Echo Request message, it MUST restart its Hello timer (section 3.3.2.3) and respond with an Echo Response message.

3.1.5.3.8 Receiving an Echo Response Message

If the SSTP client receives an Echo Response message in response to the Echo Request message it sent, it MUST restart its Hello timer. The client MUST disconnect the connection if it does not receive an Echo Response (or any other SSTP packet) within the next Hello timer interval. If the client receives an Echo Response message but did not send an Echo Request message, it SHOULD restart its Hello timer.

3.1.6 Timer Events

3.1.6.1 Negotiation Timer Processing

When a negotiation timer expires, the SSTP peer MUST send a Call Abort message and start the process of bringing down (disconnecting) the connection.

3.1.7 Other Local Events

See section 3.3.7.

3.2 Server Details

The following section provides details of the SSTP server.

3.2.1 Abstract Data Model

See section 3.3.1.

3.2.1.1 State Machine

3.2.1.1.1 Call Establishment

The following figure shows the state machine when the client establishes the outgoing SSTP tunnel.

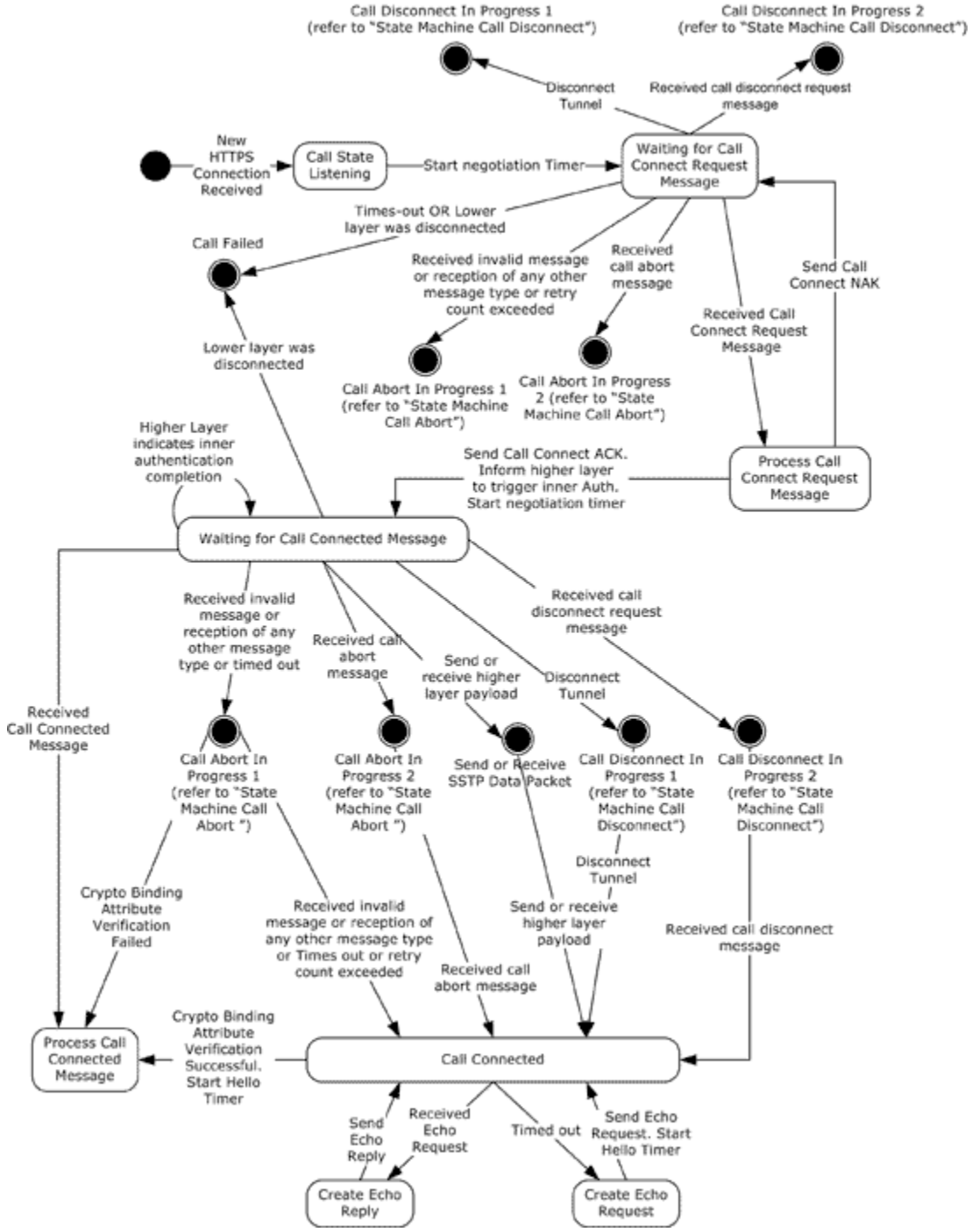


Figure 3: Server call establishment

3.2.2 Timers

Common timers are specified in section 3.3.2.

3.2.2.1 Negotiation Timer

When establishing the SSTP connection, the SSTP server starts the negotiation timer after sending the Call Connect Acknowledge message or Call Connect Negative Acknowledgment message. If it does not receive a Call Connected message or a Call Connect Request message within the timer expiry, it MUST send a Call Abort message and start the process of bringing down (disconnecting) the connection. The server MAY implement different values of timer for the Call Connected message and the Call Connect Request message.

This timer SHOULD be set to 60 seconds.<7>

3.2.3 Initialization

Server initialization MAY<8> be performed when the SSTP server software is started or when the administrator configures the SSTP server software. When the server is initialized, it MUST start a listener to listen for HTTPS requests on a predefined URI that will be used by the client<9> and the server state machine waits for an incoming HTTPS connection, as shown in the above figure, "Server call establishment," in section 3.2.1.1.1.

3.2.4 Higher-Layer Triggered Events

Events MAY be triggered from the higher layer on the server to accept or disconnect SSTP requests.

When the server is configured to accept SSTP connections, the SSTP layer is notified. The SSTP layer sets up the HTTPS layer to accept connections that request SSTP. This process allows the server to receive requests for SSTP and process them.

The server MAY request to disconnect a connection. These requests usually occur due to idle time-out or policy requirements. Server-side disconnect processing is the same as client-side disconnect processing. For more information, see section 3.2.5.2.5.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Status and Error Handling

See section 3.3.5.1.

3.2.5.2 Packet Processing

3.2.5.2.1 General Packet Validation

When the SSTP peer receives a packet, it MUST validate that the packet conforms to the syntax as specified in sections 2.2.1 and 2.2.13. If an invalid packet is received, error handling MUST occur. For more information about error handling, see section 3.3.5.1.

3.2.5.2.2 Receiving a Call Connect Request Message

The Call Connect Request message MUST be either the first message that is received by an SSTP server, or it must be received in response to the Call Connect Negative Acknowledgment message

that is sent by the server. The Call Connect Request message MUST contain an Encapsulated Protocol ID attribute with Protocol ID = PPP. For more information about PPP, see [RFC1661].

If the packet is acceptable to the server, the server SHOULD receive the PPP control frames from the client and request the PPP layer to start the FSM. The server MUST start the negotiation timer, and it MUST respond with a Call Connect Acknowledge message.

If an attribute is not acceptable in the Call Connect Request message, the server MUST respond with a Call Connect Negative Acknowledgment message that contains the Status Info attributes for all attributes that are not acceptable to the server. The server SHOULD maintain an implementation-dependent retry counter<10> that limits the number of times it sends a NAK to an unacceptable Call Connect Request message. If this count is exceeded, the server MUST send a Call Abort message and start the process of bringing down (disconnecting) the connection. If an unacceptable packet is received, error handling MUST occur. For error handling, see section 3.3.5.1.

3.2.5.2.3 Receiving a Call Connected Message

The Call Connected message MUST be received by an SSTP server only in response to a Call Connect Acknowledge message that is sent by the SSTP server before the negotiation timer expires. The server MUST verify the following:

- A Crypto Binding attribute is present.
- The Crypto Binding contains the same nonce that it sent in the Call Connect Acknowledge message.
- The Crypto Binding contains the hash of the certificate that the server used for establishing the HTTPS connection. (If a load balancer is in front of the SSTP server, the server MUST validate that the hash is the same as the configured value on the SSTP server.)
- The Crypto Binding contains a hash protocol that is supported by the server (that is, it uses one of the hash protocols that it proposed in the Call Connect Acknowledge message).
- The server MUST verify that the binding attribute contains a valid Compound MAC.

If any of the preceding checks fail, the server MUST start the abort processing procedure by sending a Call Abort message. If an unacceptable packet is received, error handling SHOULD occur. For error handling, see section 3.3.5.1. For security considerations, see section 5.

Any implementation-specific mechanism can be used by the server implementation to determine that there is a load balancer in front of the SSTP server.

The server SHOULD<11> allow the PPP data frames to pass through only if the server has received a valid Call Connected message from the client in response to its Call Connect Acknowledge message. Until a valid Call Connected message is received, the server MUST allow only PPP control frames to flow through. For more information about PPP, see [RFC1661].

3.2.5.2.4 Receiving a Call Abort Message

If the Call Abort message is received in response to the Call Abort message that is sent by the SSTP server, the server MUST clean up the connection immediately. If the SSTP server did not send a Call Abort message, the server MUST respond with a Call Abort message, wait for a time TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.1), and then clean up the connection.

3.2.5.2.5 Receiving a Call Disconnect Message

If the SSTP server receives a Call Disconnect message, the server MUST respond with a Call Disconnect Acknowledge message, wait for a time period of length TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.2), and then clean up the connection.

3.2.5.2.6 Receiving a Call Disconnect Acknowledge Message

If the Call Disconnect Acknowledge message is received in response to the Call Disconnect message that is sent by the SSTP server, the server MUST clean up the connection immediately. If the SSTP server did not send a Call Disconnect message, the client MUST respond with a Call Abort message, wait for a time TIMER_VAL_ABORT_STATE_TIMER_2 (as specified in section 3.3.2.1), and then clean up the connection.

3.2.5.2.7 Receiving an Echo Request Message

If the SSTP server receives an Echo Request message, it MUST restart its Hello timer (section 3.3.2.3) and respond with an Echo Response message.

3.2.5.2.8 Receiving an Echo Response Message

If the SSTP server receives an Echo Response message in response to the Echo Request message it sent, it MUST restart its Hello timer. The server MUST disconnect the connection if it does not receive the Echo Response message (or any other SSTP packet) during the next Hello timer period. If the server did not send an Echo Request message, it SHOULD restart its Hello timer.

3.2.6 Timer Events

3.2.6.1 Negotiation Timer Processing

On negotiation timer expiry, the SSTP peer MUST send a Call Abort message and start the process of bringing down (disconnecting) the connection.

3.2.7 Other Local Events

See section 3.3.7.

3.3 Common Details

The following details are common between the SSTP client and the SSTP server.

3.3.1 Abstract Data Model

This section describes a model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with what is described in this document.

3.3.1.1 State Machine

This section describes the state machine that applies to both the client and server for the Call Disconnect and Call Abort phases. The state machine for call establishment is different for the client and server, and is as specified in sections 3.1.1.1.1 and 3.2.1.1.1, respectively.

3.3.1.1.1 State Machine Call Disconnect

The following figure gives the state machine when the call is disconnected because of a disconnection request that is received from a higher layer or from the SSTP far end.

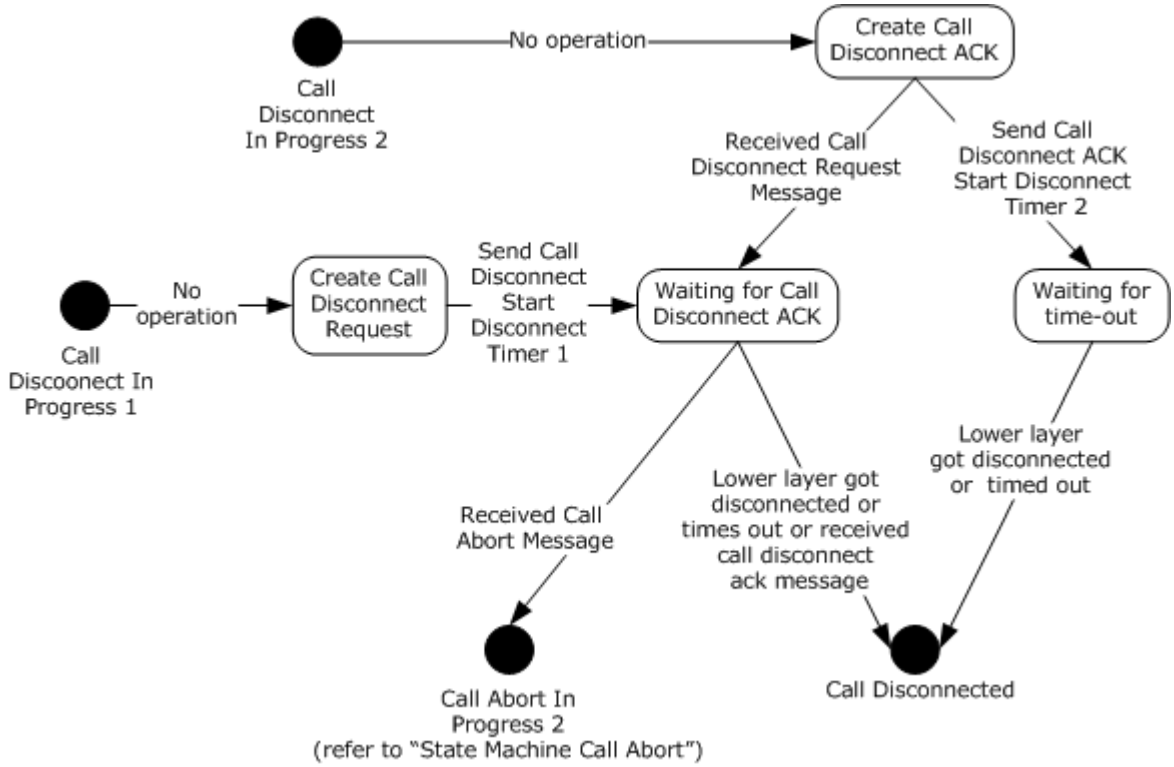


Figure 4: Common details for call disconnect

3.3.1.1.2 State Machine Call Abort

The following figure gives the state machine when a call is aborted.

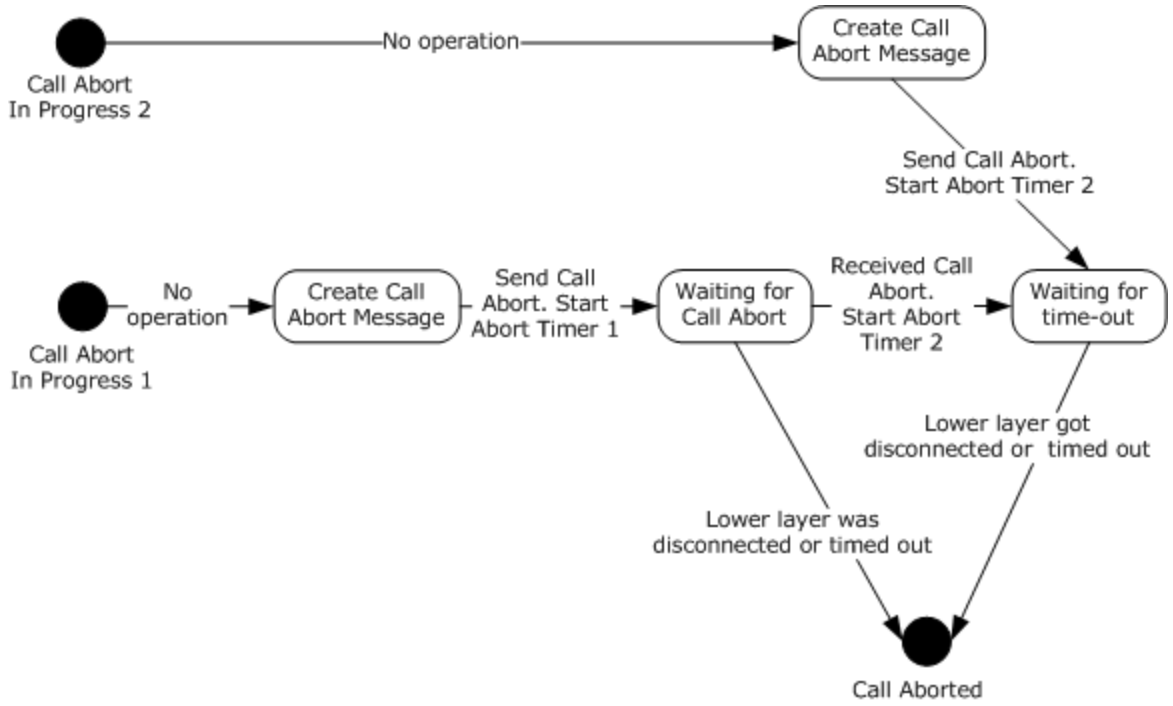


Figure 5: Common details for call abort

3.3.2 Timers

3.3.2.1 Abort-Related Timers

There are two timers related to abort processing:

- TIMER_VAL_ABORT_STATE_TIMER_1
- TIMER_VAL_ABORT_STATE_TIMER_2

The first timer is started by an SSTP peer when it has initiated an abort procedure by sending the Call Abort message to an SSTP far end. If the initiating SSTP peer receives a Call Abort message from the SSTP far end before the timer expires, it MUST cancel the timer and immediately perform a cleanup of the connection.

When the SSTP far end receives a Call Abort message, it responds with a Call Abort message after starting the second timer (that is, TIMER_VAL_ABORT_STATE_TIMER_2). This timer ensures that the SSTP peer receives the Call Abort message that is sent by the SSTP far end. This is to address collision-type situations where the SSTP peer and the SSTP far end initiate the Call Abort message at the same time (and not in response to the Call Abort message initiated by the SSTP peer). This short delay ensures that both the SSTP peer and the SSTP far end receive the Call Abort message that is sent by one another.

The first timer SHOULD be set at 3 seconds. The second timer SHOULD be set to 1 second.

3.3.2.2 Disconnect-Related Timers

There are two timers related to disconnect processing:

- TIMER_VAL_DISCONNECT_STATE_TIMER_1
- TIMER_VAL_DISCONNECT_STATE_TIMER_2

The first timer is started by an SSTP peer when it has initiated a disconnect by sending the Call Disconnect message to the SSTP far end. If the SSTP peer receives a Call Disconnect Acknowledge message from the SSTP far end before the timer expires, it MUST cancel the timer and immediately perform a cleanup of the connection.

When the SSTP far end receives a Call Disconnect message, it responds with a Call Disconnect Acknowledge message after it starts the second timer (that is, TIMER_VAL_DISCONNECT_STATE_TIMER_2). This timer ensures that the SSTP peer receives the Call Disconnect Acknowledge message that is sent by the SSTP far end.

The first timer SHOULD be set to 5 seconds. The second timer SHOULD be set to 1 second.

3.3.2.3 Hello Timer

To detect HTTPS connection failures in the absence of any active data transfers, Echo Request and Echo Response messages are periodically sent by the SSTP peer and SSTP far end, respectively. Both the SSTP client and SSTP server start a Hello timer after a successful SSTP connection setup. For example, after a successful Call Connect Request, Call Connect Acknowledge, and Call Connected message exchange, a Hello timer is initiated. The timer is restarted after receiving an SSTP packet (both the SSTP control packet and the SSTP data packet).

On receipt, the SSTP far end MUST respond with an Echo Response packet. If the SSTP far end does not respond with an Echo Response (or any SSTP packet) within the next timer interval, it MUST abort the connection without sending a Call Abort packet.

The Hello timer SHOULD be set to 60 seconds.

3.3.3 Initialization

See sections 3.1.3 and 3.2.3.

3.3.4 Higher-Layer Triggered Events

See sections 3.1.4 and 3.2.4.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Status and Error Handling

SSTP MUST attempt to delineate the SSTP frames that are contained in the HTTPS stream. If the delineation fails because of an unsupported or unrecognized packet format, the connection MUST be forcefully aborted without any SSTP message being sent to the SSTP far end. This is done to avoid the use of unreliable streams.

For the attribute parameters that are received from the Call Connect Request message, the server MAY respond with a negative acknowledgment. In the negative acknowledgment message, the server MUST include the list of attributes that are not acceptable and the reason they were not accepted.

For a list of status acknowledgments, see section 2.2.8.

3.3.5.2 SSTP Packet Processing

SSTP packet processing is covered in sections 3.1.5.3 and 3.2.5.2.

3.3.6 Timer Events

3.3.6.1 Abort Timer Processing

When the TIMER_VAL_ABORT_STATE_TIMER_1 expires, the SSTP peer MUST immediately perform forceful cleanup of the connection.

When the TIMER_VAL_ABORT_STATE_TIMER_2 expires, the SSTP peer MUST immediately clean up the connection.

3.3.6.2 Disconnect Timer Processing

When the TIMER_VAL_DISCONNECT_STATE_TIMER_1 expires, the SSTP peer MUST immediately perform forceful cleanup of the connection.

When the TIMER_VAL_DISCONNECT_STATE_TIMER_2 expires, the SSTP peer MUST immediately clean up the connection.

3.3.6.3 Hello Timer Processing

When the Hello timer expires, an Echo Request packet MUST be sent by the SSTP peer to the SSTP far end.

3.3.7 Other Local Events

The SSTP server receives notification from the lower layer when a new incoming HTTPS connection is established.

The SSTP peer may receive local events because of network interface failure, network failure, TCP failure, SSL/TLS failure, and similar scenarios. In all such scenarios, the SSTP layer MUST immediately clean up the call-related information without any over-the-wire interaction, and MUST indicate the same to the higher layers (PPP) that are present. For more information about PPP, see [RFC1661].

4 Protocol Examples

The following sections describe operations as used in common scenarios to illustrate the function of the Secure Socket Tunneling Protocol (SSTP).

4.1 HTTPS Layer Establishment

During initialization, the SSTP server configures both the server certificate to use and the URL in which it is interested. This URL will be a well-known URL between the client and the server. The server may also support HTTP to allow SSL/TLS terminating edge devices.<12> These devices terminate the SSL/TLS connection coming from the client, validate the URL, and establish the HTTP connection to the actual Web server behind it.

The request sent to the SSTP server uses the HTTP verb SSTP_DUPLEX_POST with content length encoding.

The request sent is as follows:

- Method: SSTP_DUPLEX_POST
- URI: /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/
- Protocol Version: HTTP/1.1
- Content-Length: 18446744073709551615 (ULONGLONG_MAX)
- Host: <Server Name>

As a part of setting up a bidirectional session with HTTPS, when the HTTP request is being sent, no entity body message will be sent to the far end. Instead, the client initiates a timer (for 60 seconds) and sends out the request to the server. A response is expected within 60 seconds. The server will be listening for the URI /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/. The SSTP server, on receiving the request, validates the method to be SSTP_DUPLEX_POST and the HTTP version to be 1.1. If this succeeds, and there are sufficient ports on the server to accept the new connection, the server sends back an HTTP_STATUS_OK message to the client. Otherwise, the server fails the request by sending an HTTP error code containing indication this is to be the last data being sent over the connection.

The response sent is as follows:

```
Protocol Version: HTTP/1.1
Status code: 200
Content-Length: 18446744073709551615
Server: Microsoft-HTTPAPI/2.0
Date: Thu, 09 Nov 2006 00:51:09 GMT
```

4.2 HTTP Layer Teardown

After the SSTP finite state machine (FSM) is completed, it performs its own teardown. When the teardown is completed, the SSTP FSM signals the HTTP layer to tear down itself.

The teardown of the HTTP connection layer is done by signaling the request/response completion. The client closes the request to indicate that no data transfer is expected on it. If the server is initiating the disconnect, it indicates the end of the entity body to the HTTP layer. The HTTP layer closes the TCP connection appropriately.

4.3 SSTP Layer Establishment

After the bidirectional HTTPS layer is up, the SSTP finite state machine (FSM) begins. The server initializes the FSM by waiting for the SSTP Call Connect Request message to arrive from the client. After the Call Connect Request message arrives, the server validates the Encapsulated Protocol ID attribute value for PPP. The SSTP server then responds to the client by using the Call Connect Acknowledge message. The client sends the Call Connected message to the server to indicate that it is ready to send data traffic. When the server receives the Call Connected message, it allows bidirectional data transfer on the SSTP connection.

After the control channel is established, the echo timers continue to send Echo Request messages in order to keep the channel alive. Echo Responses are sent and received to ensure that the connection is not broken.

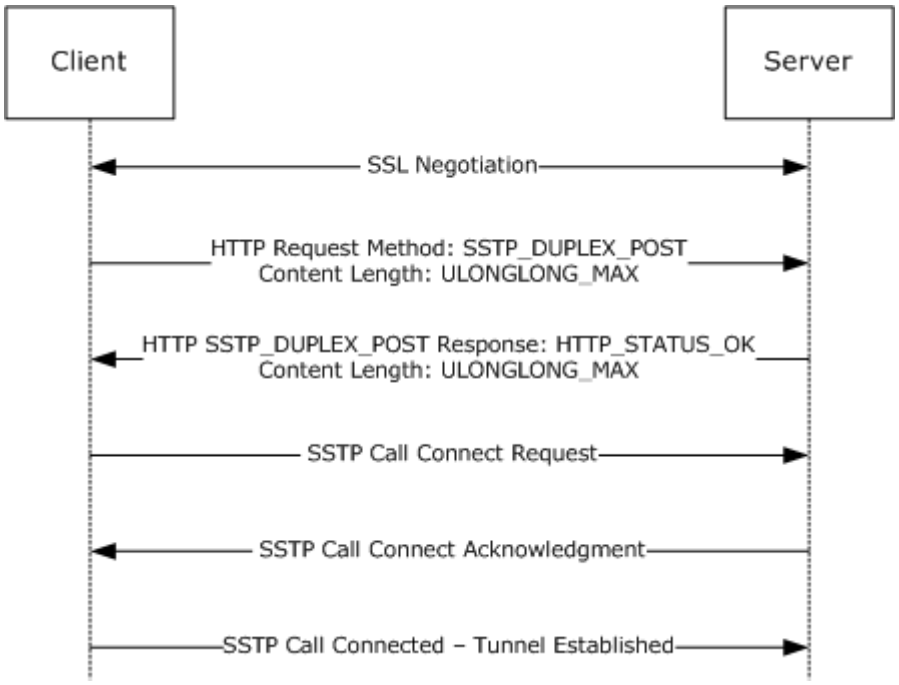


Figure 6: SSTP call setup for a non-proxy scenario

4.4 SSTP Layer Teardown

SSTP has forceful teardown through the Call Abort message as well as graceful shutdown through the Call Disconnect message.

The Call Abort message is used in situations in which the SSTP control channel negotiation has encountered a problem such as a time-out, an invalid message, or other similar problems. The Call Abort-based shutdown uses short timers for quick cleanup.

Graceful shutdown occurs when the Call Disconnect message is sent to the far end. The far end responds by sending a Call Disconnect Acknowledge message in order to signal the end of data transfer.

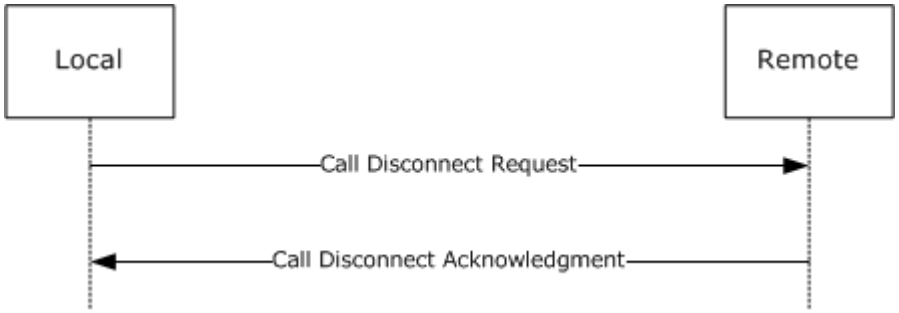


Figure 7: SSTP call teardown

4.5 Handling HTTP Proxies

In the case that a tunnel is established through a proxy, the typical CONNECT request looks like this:

```
Method: CONNECT
Protocol Version: HTTP/1.1
URI:  <Server Name>:443
Host: <Server Name>:443
SSTPVERSION: 1.0
```

The SSTPVERSION field can be used by network administrators through forward proxies to filter the SSTP-based connection to go out of the network.

The response from the proxy follows. After the response is received, the SSL/TLS traffic is sent in an encrypted manner to the proxy, and the proxy relays the encrypted traffic between the client and the server.

```
Protocol Version: HTTP/1.1
Status Code: 200 HTTP_STATUS_OK
Reason: Connection Established
Proxy-Connection: Keep-Alive
Connection: Keep-Alive
```

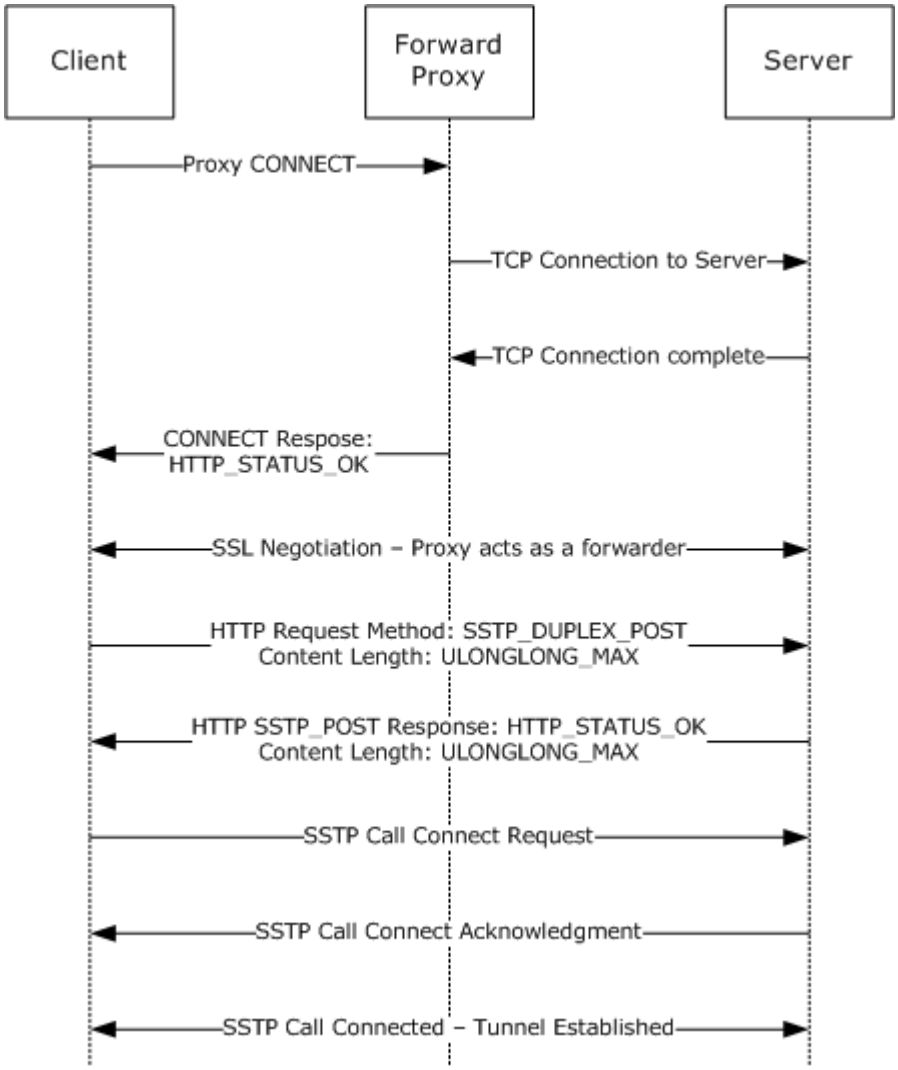


Figure 8: SSTP call setup for a proxy scenario

4.6 Crypto Binding

The client sends an SSTP_MSG_CALL_CONNECT_REQUEST that encapsulates the PPP protocol. The actual data bytes are:

```
10 01 00 0E 00 01 00 01 00 01 00 06 00 01
```

The details for the packet are:

- Version: 0x10 (Major Version: 0x1, Minor Version: 0x0)
- C: 1 (Control Packet)
- Length: 0x00E

- Message Type: 0x0001 (SSTP_MSG_CALL_CONNECT_REQUEST)
- Num Attributes: 0x0001
- Attribute 1:
  - Attribute ID: 0x01 (SSTP_ATTRIB_ENCAPSULATED_PROTOCOL_ID)
  - Length: 0x006
  - Value: 0x0001 (SSTP_ENCAPSULATED_PROTOCOL_PPP)

The server responds to the client with SSTP_MSG_CALL_CONNECT_ACK. In this case, the server supports only the SHA256 hash algorithm for crypto binding. The actual data bytes are:

```
10 01 00 30 00 02 00 01 00 04 00 28 00 00 00 02
41 2B 48 9A EB D7 EC C7 D0 89 66 F2 6B E7 CD 72
B2 31 A0 E9 21 0D 7C 91 B3 08 86 2B 03 44 C4 35
```

The details are:

- Version: 0x10 (Major Version: 0x1, Minor Version: 0x0)
- C: 1 (Control Packet)
- Length: 0x030
- Message Type: 0x0002 (SSTP_MSG_CALL_CONNECT_ACK)
- Num Attributes: 0x0001
- Attribute 1:
  - ID: 0x04 (SSTP_ATTRIB_CRYPTO_BINDING_REQ)
  - Length: 0x028
  - Value:
    - Protocol Bitmask: 0x02 (CERT_HASH_PROTOCOL_SHA256)
    - Nonce:

```
41 2B 48 9A EB D7 EC C7 D0 89 66 F2 6B E7 CD 72
B2 31 A0 E9 21 0D 7C 91 B3 08 86 2B 03 44 C4 35
```

The client continues with the PPP negotiation after receiving the preceding message. When PPP authentication is finished, the client completes the crypto binding by sending an SSTP_MSG_CALL_CONNECTED message. The data bytes that are transmitted in this scenario are:

```
10 01 00 70 00 04 00 01 00 03 00 68 00 00 00 02
41 2B 48 9A EB D7 EC C7 D0 89 66 F2 6B E7 CD 72
B2 31 A0 E9 21 0D 7C 91 B3 08 86 2B 03 44 C4 35
```

```
79 93 EF 31 4C 49 3D AC E9 F0 2D 60 E7 E6 1C 84
B6 69 0A AF E9 D7 AE EA 92 CB BE 8A D5 99 42 2D
52 A6 8E FD 8C FF BF 52 77 0B 8F 0F E8 EC 73 71
65 83 AF 6D 61 1E B6 D1 79 B3 B2 08 40 98 54 49
```

The computation of Compound MAC is done based on the following:

Higher-Layer Authentication Key (HLAK):

```
2A 1B B4 0D 55 AB 0F 5E F3 2F 06 F2 B3 CC 73 C4
8F D3 FA C4 1D 7A 13 15 A1 92 28 D9 02 4C A1 64
```

Hash of the certificate that is provided by the server:

```
79 93 EF 31 4C 49 3D AC E9 F0 2D 60 E7 E6 1C 84
B6 69 0A AF E9 D7 AE EA 92 CB BE 8A D5 99 42 2D
```

The details of the packet that is sent are:

- Version: 0x10 (Major Version: 0x1, Minor Version: 0x0)
- C: 1 (Control Packet)
- Length: 0x070
- Message Type: 0x0004 (SSTP_MSG_CALL_CONNECTED)
- Num Attributes: 0x0001
- Attribute 1:
  - Attribute ID: 0x03 (SSTP_ATTRIB_CRYPTO_BINDING)
  - Length: 0x068
  - Value:
    - Hash Protocol Bitmask: 0x02 (CERT_HASH_PROTOCOL_SHA256)
    - Nonce:

```
41 2B 48 9A EB D7 EC C7 D0 89 66 F2 6B E7 CD 72
B2 31 A0 E9 21 0D 7C 91 B3 08 86 2B 03 44 C4 35
```

    - Certificate Hash:

```
79 93 EF 31 4C 49 3D AC E9 F0 2D 60 E7 E6 1C 84
B6 69 0A AF E9 D7 AE EA 92 CB BE 8A D5 99 42 2D
```

- Compound MAC:

```
52 A6 8E FD 8C FF BF 52 77 0B 8F 0F E8 EC 73 71
65 83 AF 6D 61 1E B6 D1 79 B3 B2 08 40 98 54 49
```

In this example, the server uses a SHA1 hash for crypto binding. The following is a sample SSTP_MSG_CALL_CONNECT_ACK in this scenario:

```
10 01 00 30 00 02 00 01 00 04 00 28 00 00 00 01
0F 1A 2D 58 D4 A3 E3 00 0F AD 3C E4 90 6E 07 B7
07 AA 9E 44 1C CE AC 5C BD 7B 2C C1 C9 D8 6C DF
```

The details of the packet are:

- Version: 0x10 (Major Version: 0x1, Minor Version: 0x0)
- C: 1 (Control Packet)
- Length: 0x030
- Message Type: 0x0002 (SSTP_MSG_CALL_CONNECT_ACK)
- Num Attributes: 0x0001
- Attribute 1:
  - Attribute ID: 0x04 (SSTP_ATTRIB_CRYPTO_BINDING_REQ)
  - Length: 0x028
  - Value:
    - Hash Protocol Bitmask: 0x01 (CERT_HASH_PROTOCOL_SHA1)
    - Nonce:

```
0F 1A 2D 58 D4 A3 E3 00 0F AD 3C E4 90 6E 07 B7
07 AA 9E 44 1C CE AC 5C BD 7B 2C C1 C9 D8 6C DF
```

For this SSTP_MSG_CALL_CONNECT_ACK, the following shows a valid crypto binding completion via the SSTP_MSG_CALL_CONNECTED message.

```
10 01 00 70 00 04 00 01 00 03 00 68 00 00 00 01
0F 1A 2D 58 D4 A3 E3 00 0F AD 3C E4 90 6E 07 B7
07 AA 9E 44 1C CE AC 5C BD 7B 2C C1 C9 D8 6C DF
58 26 B6 29 BD A5 9B 8E 6F D8 DC D2 62 2F D3 4C
53 48 05 A5 00 00 00 00 00 00 00 00 00 00 00 00
```

```
69 91 5D D5 83 D8 06 2F EF 16 F6 1D B2 F0 32 90
EC 27 CB 6C 00 00 00 00 00 00 00 00 00 00 00 00
```

The compound MAC is computed based on the following values for HLAK and certificate hash:

Higher-Layer Authentication Key (HLAK):

```
4B 31 28 F4 39 25 D9 00-6E EF B1 C4 E8 65 15 A1
D8 8E 56 BA B3 CA 2B DF-03 73 B7 F5 A8 A1 3B 19
```

Hash of the certificate that is provided by the server:

```
58 26 B6 29 BD A5 9B 8E 6F D8 DC D2 62 2F D3 4C
53 48 05 A5
```

The details of the packet are:

- Version: 0x10 (Major Version: 0x1, Minor Version: 0x0)
- C: 1 (Control Packet)
- Length: 0x070
- Message Type: 0x0004 (SSTP_MSG_CALL_CONNECTED)
- Num Attributes: 0x0001
- Attribute 1:
  - Attribute ID: 0x03 (SSTP_ATTRIB_CRYPTO_BINDING)
  - Length: 0x068
  - Value:
    - Hash Protocol Bitmask: 0x01 (CERT_HASH_PROTOCOL_SHA1)
    - Nonce:

```
0F 1A 2D 58 D4 A3 E3 00 0F AD 3C E4 90 6E 07 B7
07 AA 9E 44 1C CE AC 5C BD 7B 2C C1 C9 D8 6C DF
```

    - Certificate Hash:

```
58 26 B6 29 BD A5 9B 8E 6F D8 DC D2 62 2F D3 4C
53 48 05 A5
```

- Compound MAC:

```
69 91 5D D5 83 D8 06 2F EF 16 F6 1D B2 F0 32 90
EC 27 CB 6C
```

5 Security

The following sections specify security considerations for implementers of the Secure Socket Tunneling Protocol (SSTP).

5.1 Security Considerations for Implementers

Because SSTP Version 1 only supports transport of PPP frames, there is no need for any negotiation of parameters in the SSTP Call Connect Request message, Call Connect Acknowledge message, and Call Connected message exchange. When the server receives a Call Connect Request message, it sends a Call Connect Acknowledge message and triggers the PPP state machine. When the SSTP client receives the Call Connect Acknowledge message, it triggers the PPP state machine.

The SSTP server should only begin forwarding the PPP data frames after it validates the Crypto Binding attribute in the Call Connected message from the SSTP client. The server should drop any PPP data frames that are received before the Call Connected message is received. For more information about PPP, see [RFC1661].

5.2 Index of Security Parameters

Security parameter	Section
Authentication	2.1
Hashing algorithms	3.1.5.2

5.3 Attack Scenarios

5.3.1 Unauthorized Client Connecting to an SSTP Server

In this scenario, an unauthorized attacker poses as a valid SSTP client and tries to connect to a valid SSTP server. The HTTPS connection goes through because the server does not authenticate the client at the SSL/TLS layer. The connection MUST be terminated by the SSTP server at the PPP layer after determining that the client has no proper user credentials. For more information, see [RFC1661].

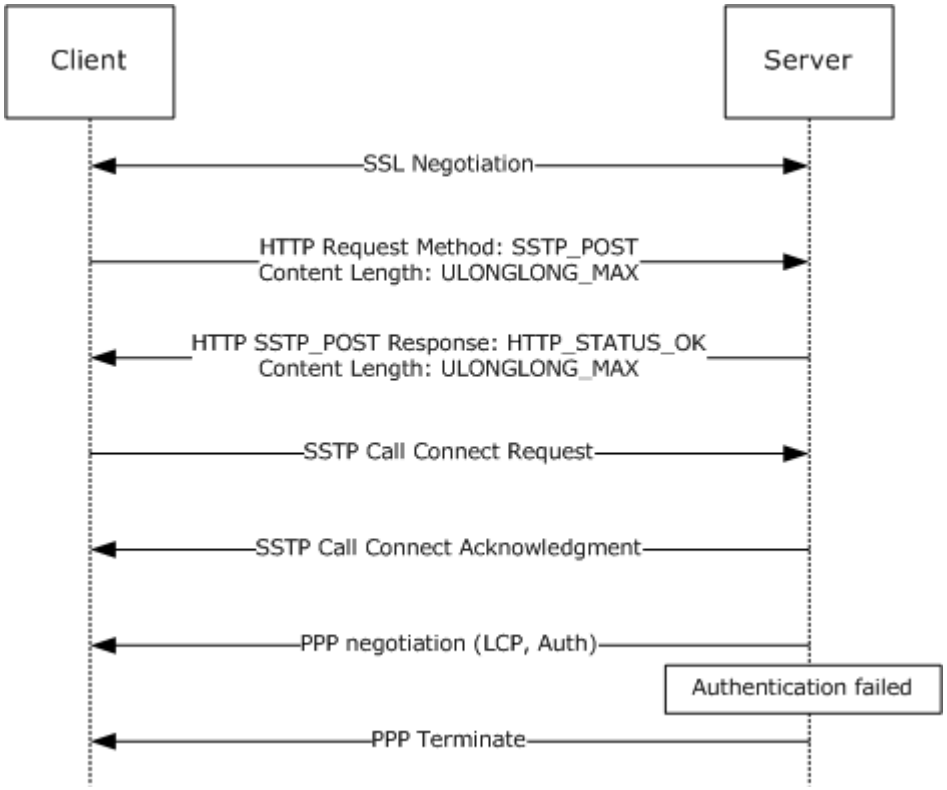


Figure 9: Unauthorized client connecting to an SSTP server

5.3.2 Unauthorized SSTP Server Accepting Connections from a Genuine SSTP Client

In this scenario, a valid SSTP client is redirected by an attacker to an unauthorized SSTP server (for example, by DNS poisoning). In this scenario, the connection is terminated by the client at the SSL/TLS layer when the certificate validation check fails. It is recommended that the SSTP client validates that the common name and subject name in the server certificate matches the host name to which the client established the connection. Also, it is recommended that the SSTP client validates that the server certificate contains either the server authentication or all purpose extended key usage (EKU).

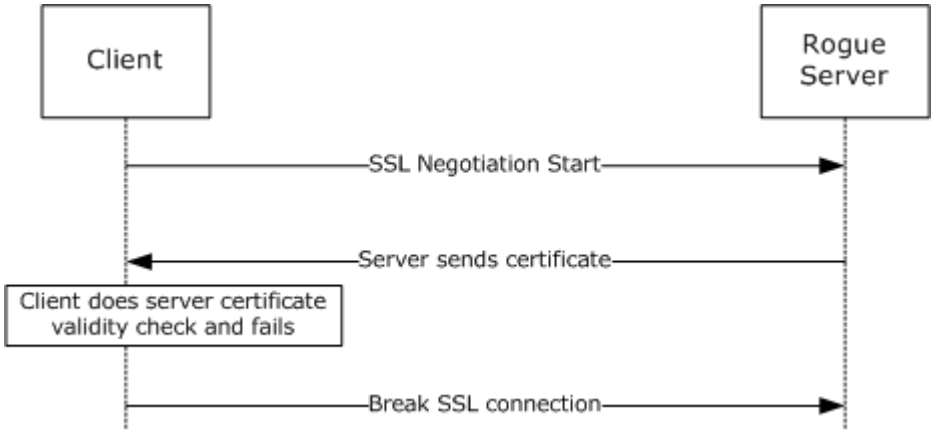


Figure 10: Client connecting to an unauthorized SSTP server

5.3.3 Man in the Middle

In this scenario, an attacker poses as a man in the middle (MITM). For example, a man in the middle may be using a rogue wireless access point in a wireless-enabled enterprise environment.

The data flow in case of attack (without an SSTP crypto binding solution) looks like this:

- The man in the middle establishes an HTTPS connection with the SSTP server.
- By using some technique (such as a rogue access point (AP) that has a similar name to the enterprise network), the man-in-the-middle attacker gets a real client to initiate an EAP authentication (which can be any EAP method) with an authorized SSTP server. The client cannot determine that the HTTPS channel has been established to the man-in-the-middle machine; the client attempts to authenticate to a known authorized server by using EAP authentication, as usual.
- The man in the middle passes (or re-routes) the client's EAP-TLS authentication packets that are received over wireless to the PPP over SSTP (over SSL/TLS) tunnel it has established with the SSTP server. It does the same thing in reverse for responses to the client.
- The client and the server successfully complete the EAP authentication. The man-in-the-middle machine simply relays the packets back and forth between both SSL/TLS tunnels.
- The man in the middle drops the client and continues to use the authenticated SSTP channel established with the server—without knowing the client's privileges and in an unauthorized manner.

Note The above attack can happen for any PPP authentication protocol that can be relayed on another transport. For example, EAP can be relayed on SSTP as well as wireless.

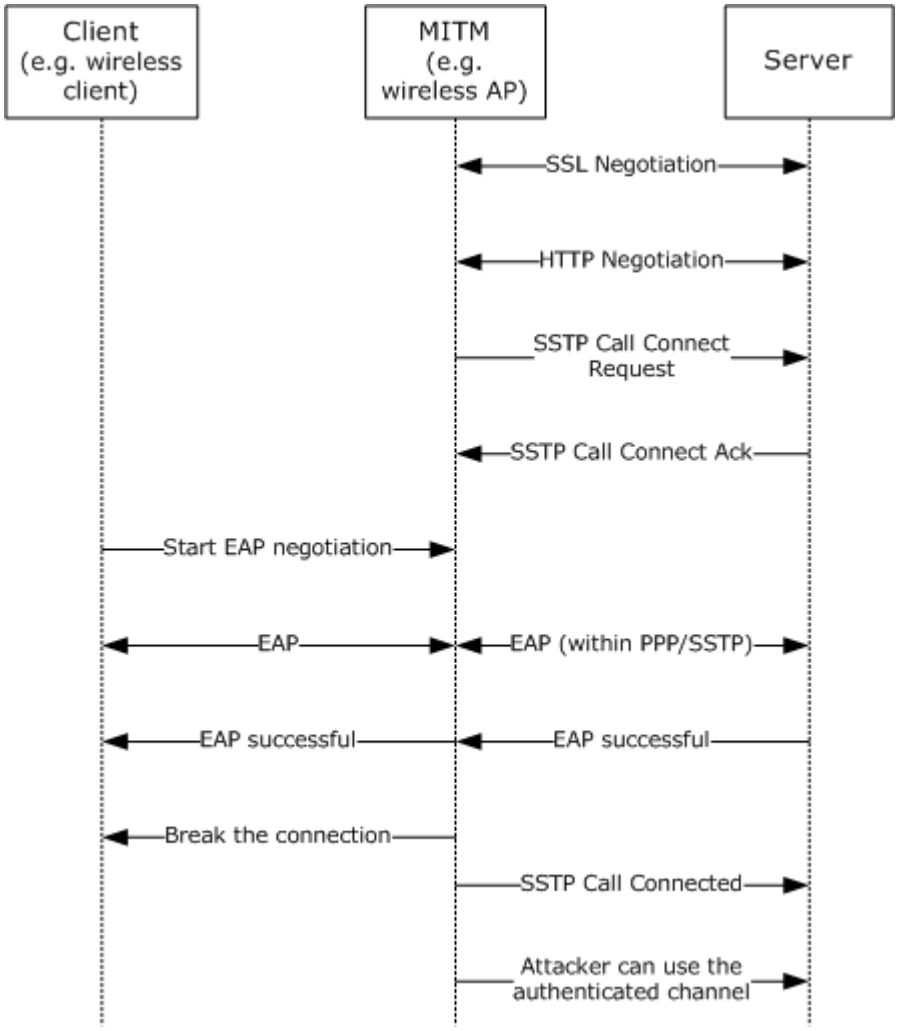


Figure 11: MITM scenario without the SSTP crypto binding solution

To mitigate this attack, the SSTP server expects a Crypto Binding attribute from the SSTP client to be present in the Call Connected message. This attribute is generated by the client using the keys generated on the client. By using the inner (or PPP) authentication phase keys, and by tying the inner (or PPP) authentication to the outer (or SSL/TLS) authentication phase, this technique ensures that the SSTP client and the SSTP server participated in the inner authentication and terminate at the expected endpoints.

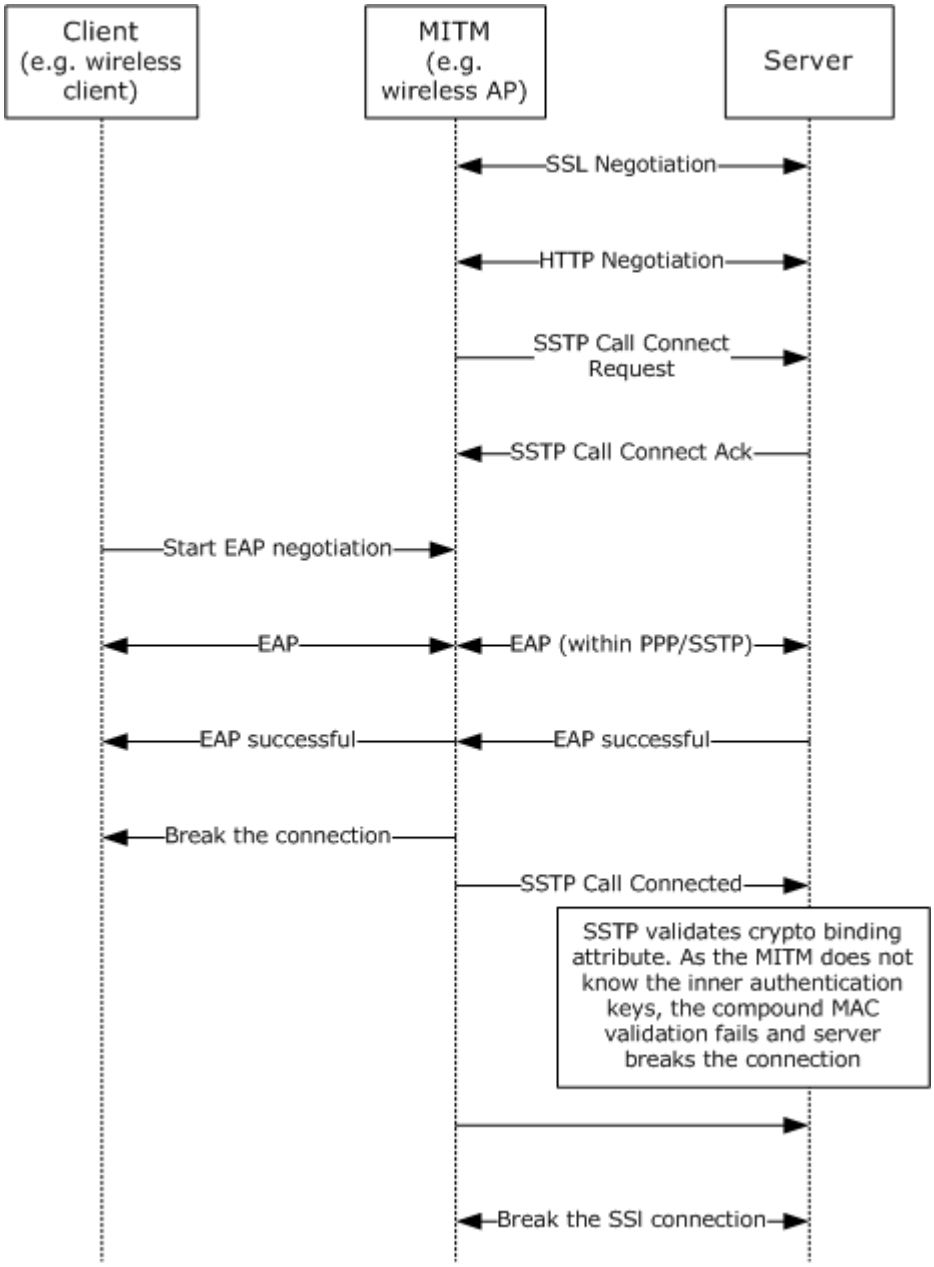


Figure 12: MITM scenario with SSTP crypto binding solution

Note Protected EAP (PEAP) (for more information, see [MS-PEAP]) can also be used as the authentication protocol. In this case, the security attack vector and the solution remain the same as EAP. PEAP has an outer TLS channel between the PEAP client and authenticating server (like radius server) and does inner EAP authentication. If a PEAP crypto Type-Length-Value (TLV) check is not enabled on the client, the PEAP client is susceptible to PEAP man-in-the-middle attacks. SSTP does not offer a solution to this attack vector, which is already solved by the PEAP crypto TLV attribute.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Server 2008
- Windows Vista

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification that is prescribed by using the terms SHOULD or SHOULD NOT implies Windows behavior in accord with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 2.1: Windows Server 2008 does not support HTTPS client authentication. Windows Server 2008 supports client authentication by using MS-CHAPv2, EAP-TLS, PEAP-MSCHAPv2 and PEAP-TLS. Windows Server 2008 also supports client authentication by using PAP and CHAP but does not recommend their usage due to security reasons.

<2> Section 2.2.8: Windows Server 2008 allows a retry count of 3.

<3> Section 2.2.13: A Windows implementation always sends a Status Info attribute in a Call Abort message and Call Disconnect message.

<4> Section 3.1.2.1: The Windows-based client starts a timer with a value of 60 seconds after sending a Call Connected message and starts a timer with a value of 60 seconds after receiving a Call Connected message.

<5> Section 3.1.4.1: By default, Windows uses URI/sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/. The Windows-based client sends a content length of ULONGLONG_MAX (18446744073709551615). The Windows-based client validates that the common name or subject name in the server certificate is the same as the host name to which the connection is being established. The Windows-based client also validates that the server certificate contains either server authentication or All Purpose extended key usage (EKU).

<6> Section 3.1.5.3.3: The Windows client retries 3 times.

<7> Section 3.2.2.1: Windows Server 2008 waits 60 seconds for the Call Connected message and 60 seconds for the Call Connect Request message.

<8> Section 3.2.3: Windows Server 2008 initializes the server state when Routing and Remote Access is started. It also initializes the server state when SSTP ports are configured.

<9> Section 3.2.3: By default, Windows uses the URI: /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/.

<10> Section 3.2.5.2.2: Windows Server 2008 allows a retry count of 3.

<11> Section 3.2.5.2.3: Windows Server 2008 starts allowing PPP control frames from the client and requests the PPP layer to start the FSM. However, it does not allow any data frames until the PPP negotiation is completed.

<12> Section 4.1: By default, the Windows implementation supports only HTTPS traffic. HTTP may be enabled via a registry key.

7 Index

A

Abort timers (section 3.3.2.1, section 3.3.6.1)
Abstract data model
  client (section 3.1.1, section 3.3.1)
  server (section 3.2.1, section 3.3.1)
Applicability
Attack scenarios - security

C

Call abort - state machine
Call abort message
  client
  server
Call connect acknowledgment message - client
Call connect message - server
Call connect negative acknowledgment message - client
Call connected message - server
Call disconnect - state machine
Call disconnect acknowledgment message
  client
  server
Call disconnect message
  client
  server
Call establishment
  client
  server
Call_Connect_Acknowledge_Message packet
Call_Connect_Negative_Acknowledgment_Message packet
Call_Connect_Request_Message packet
Call_Connected_Message packet
Capability negotiation
Client
  abstract data model (section 3.1.1, section 3.3.1)
  call abort message
  call connect acknowledgment message
  call connect negative acknowledgment message
  call disconnect acknowledgment message
  call disconnect message
  call establishment
  crypto binding
  Echo Request message
  Echo Response message
  error handling (section 3.1.5.1, section 3.3.5.1)
  higher-layer triggered events (section 3.1.4, section 3.3.4)
  initialization (section 3.1.3, section 3.3.3)
  local events (section 3.1.7, section 3.3.7)
  message processing (section 3.1.5, section 3.3.5)
  message status (section 3.1.5.1, section 3.3.5.1)
  negotiation timer (section 3.1.2.1, section 3.1.6.1)
  overview (section 3.1, section 3.3)
  packet processing
  packet validation
  sequencing rules (section 3.1.5, section 3.3.5)
  SSTP packet processing
  state machine (section 3.1.1.1, section 3.3.1.1)
  timer events (section 3.1.6, section 3.3.6)
  timers (section 3.1.2, section 3.3.2)
Crypto binding
  client
  example
Crypto_Binding_Attribute packet
Crypto_Binding_Request_Attribute packet

D

Data model - abstract
  client (section 3.1.1, section 3.3.1)
  server (section 3.2.1, section 3.3.1)
Disconnect timers (section 3.3.2.2, section 3.3.6.2)

E

Echo Request message
  client
  server
Echo Response message
  client
  server
Encapsulated_Protocol_ID_Attribute packet
Error handling
  client (section 3.1.5.1, section 3.3.5.1)
  server (section 3.2.5.1, section 3.3.5.1)
Examples
  crypto binding
  handling HTTP proxies
  HTTPS layer establishment
  HTTPS layer teardown
  overview
  SSTP layer establishment
  SSTP layer teardown

F

Fields - vendor-extensible

G

Glossary

H

Handling HTTP proxies example
Hello timer (section 3.3.2.3, section 3.3.6.3)
Higher-layer triggered events
  client (section 3.1.4, section 3.3.4)
  server (section 3.2.4, section 3.3.4)
HMAC-SHA1-160
  input data
  key
HMAC-SHA256-256
  input data
  key

HTTPS layer establishment example
HTTPS layer teardown example

I

Implementer - security considerations
Informative references
Initialization
  client (section 3.1.3, section 3.3.3)
  server (section 3.2.3, section 3.3.3)
Input data
  crypto binding HMAC-SHA1-160
  crypto binding HMAC-SHA256-256
Introduction

K

Key
  crypto binding HMAC-SHA1-160
  crypto binding HMAC-SHA256-256

L

Local events
  client (section 3.1.7, section 3.3.7)
  server (section 3.2.7, section 3.3.7)

M

Man in the middle attack - security
Message processing
  client (section 3.1.5, section 3.3.5)
  server (section 3.2.5, section 3.3.5)
Messages
  overview
  status - client (section 3.1.5.1, section 3.3.5.1)
  status - server (section 3.2.5.1, section 3.3.5.1)
  syntax
  transport
MITM attack - security

N

Negotiation timer
  client (section 3.1.2.1, section 3.1.6.1)
  server (section 3.2.2.1, section 3.2.6.1)
Normative references

O

Overview (synopsis)

P

Packet processing
  client
  server
Packet validation
  client
  server
Parameters - security
Preconditions
Prerequisites

R

References
  informative
  normative
  overview
Relationship to other protocols
Rogue client - security
Rogue SSTP server and genuine SSTP client - security

S

Security
  attack scenarios
  implementer considerations
  MITM attack
  overview
  parameters
  rogue client
  rogue SSTP server and genuine SSTP client
Sequencing rules
  client (section 3.1.5, section 3.3.5)
  server (section 3.2.5, section 3.3.5)
Server
  abstract data model (section 3.2.1, section 3.3.1)
  call abort message
  call connect message
  call connected message
  call disconnect acknowledgment message
  call disconnect message
  call establishment
  Echo Request message
  Echo Response message
  error handling (section 3.2.5.1, section 3.3.5.1)
  higher-layer triggered events (section 3.2.4, section 3.3.4)
  initialization (section 3.2.3, section 3.3.3)
  local events (section 3.2.7, section 3.3.7)
  message processing (section 3.2.5, section 3.3.5)
  message status (section 3.2.5.1, section 3.3.5.1)
  negotiation timer (section 3.2.2.1, section 3.2.6.1)
  overview (section 3.2, section 3.3)
  packet processing
  packet validation
  sequencing rules (section 3.2.5, section 3.3.5)
  SSTP packet processing
  state machine (section 3.2.1.1, section 3.3.1.1)
  timer events (section 3.2.6, section 3.3.6)
  timers (section 3.2.2, section 3.3.2)
SSTP layer establishment example
SSTP layer teardown example
SSTP packet processing
SSTP_Attributes packet
SSTP_Control_Packet packet
SSTP_Data_Packet packet
SSTP_MSG_CALL_DISCONNECT_OR_SSTP_MSG_CALL_DISCONNECT_ACK_OR_SSTP_MSG_ECHO_REQUEST_OR_SSTP_MSG_ECHO_RESPONSE packet

SSTP_Packet packet
Standards assignments
State machine
  client (section 3.1.1.1, section 3.3.1.1)
  server (section 3.2.1.1, section 3.3.1.1)
Status - message
  client (section 3.1.5.1, section 3.3.5.1)
  server (section 3.2.5.1, section 3.3.5.1)
Status_Info_Attribute packet
Syntax

T

Timer events
  client (section 3.1.6, section 3.3.6)
  server (section 3.2.6, section 3.3.6)
Timers
  client (section 3.1.2, section 3.3.2)
  server (section 3.2.2, section 3.3.2)
Transport
Triggered events - higher-layer
  client (section 3.1.4, section 3.3.4)
  server (section 3.2.4, section 3.3.4)
Tunnel event
  disconnect
  establish

V

Vendor-extensible fields
Versioning

W

Windows behavior