Transport layer and authentication protocol learning symbols used by the learner. For each symbol, the table lists the name of the symbol without the `MSG_` prefix, the specification name without the `SSH_MSG_` prefix, the message ID for the corresponding message type, a reference to the relevant section of the specification, the message class, whether the message is sent by client or server, and additional notes such as the service or authentication method name included in the message. The different message classes are described in Section 3.3 of the paper, referred to as *S* for static, *C* for cryptographic, and *V* for variable messages.

					Send by		
Symbol	Message Type	ID	Reference	Class	C	S	Note
DEBUG	DEBUG	4	[8, Section 11.3]	V	●	●	
DISCONNECT	DISCONNECT	1	[8, Section 11.1]	V	●	●	
EXT_INFO	EXT_INFO	7	[1, Section 2.3]	V	●	●	
IGNORE	IGNORE	2	[8, Section 11.2]	V	●	●	
KEX_DH_GEX_GROUP	KEX_DH_GEX_GROUP	31	[3, Section 3]	C	○	●	
KEX_DH_GEX_INIT	KEX_DH_GEX_INIT	32	[3, Section 3]	C	●	○	
KEX_DH_GEX_OLD_REQUEST	KEX_DH_GEX_REQUEST_OLD	30	[3, Section 5]	V	●	○	
KEX_DH_GEX_REPLY	KEX_DH_GEX_REPLY	33	[3, Section 3]	C	○	●	
KEX_DH_GEX_REQUEST	KEX_DH_GEX_REQUEST	34	[3, Section 3]	V	●	○	
KEX_ECDH_INIT	KEX_ECDH_INIT	30	[12, Section 4]	C	●	○	
KEX_ECDH_REPLY	KEX_ECDH_REPLY	31	[12, Section 4]	C	○	●	
KEX_HBR_INIT	KEX_HYBRID_INIT	30	[5, Section 2.1]	C	●	○	
KEX_HBR_REPLY	KEX_HYBRID_REPLY	31	[5, Section 2.1]	C	○	●	
KEX_RSA_DONE	KEXRSA_DONE	32	[4, Section 4]	C	○	●	
KEX_RSA_PUBKEY	KEXRSA_PUBKEY	30	[4, Section 4]	C	○	●	
KEX_RSA_SECRET	KEXRSA_SECRET	31	[4, Section 4]	C	●	○	
KEXDH_INIT	KEXDH_INIT	30	[8, Section 8]	C	●	○	
KEXDH_REPLY	KEXDH_REPLY	31	[8, Section 8]	C	○	●	
KEXINIT	KEXINIT	20	[8, Section 7.1]	V	●	●	
NEWCOMPRESS	NEWCOMPRESS	8	[1, Section 3.2]	S	●	○	
NEWKEYS	NEWKEYS	21	[8, Section 7.3]	S	●	●	
PING_OPENSSH	PING	192	[10, Section 1.9]	V	●	◐	Vendor extension
PONG_OPENSSH	PONG	193	[10, Section 1.9]	V	◐	●	Vendor extension
SERVICE_ACCEPT	SERVICE_ACCEPT	6	[8, Section 10]	V	○	●	
SERVICE_REQUEST_CONNECTION	SERVICE_REQUEST	5	[8, Section 10]	S	●	○	Service `ssh-connection`
SERVICE_REQUEST_USERAUTH	SERVICE_REQUEST	5	[8, Section 10]	S	●	○	Service `ssh-userauth`
UNIMPLEMENTED	UNIMPLEMENTED	3	[8, Section 11.4]	V	●	●	
UNKNOWN_ID_ALGORITHM_NEGOTIATION	*n/a*	22	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_KEY_EXCHANGE_SPECIFIC	*n/a*	49	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_RESERVED_0	*n/a*	0	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_TRANSPORT_GENERIC	*n/a*	9	*n/a*	S	○	○	Message ID only
VERSION_EXCHANGE	*n/a*	*n/a*	[8, Section 4.2]	V	●	●	As binary packet
UNKNOWN_ID_USERAUTH_GENERIC	*n/a*	54	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_USERAUTH_SPECIFIC	*n/a*	79	*n/a*	S	○	○	Message ID only
USERAUTH_USERAUTH_BANNER	USERAUTH_BANNER	53	[6, Section 5.4]	V	○	●	
USERAUTH_FAILURE	USERAUTH_FAILURE	51	[6, Section 5.1]	V	○	●	
USERAUTH_INFO_REQUEST	USERAUTH_INFO_REQUEST	60	[2, Section 3.2]	V	○	●	
USERAUTH_INFO_RESPONSE	USERAUTH_INFO_RESPONSE	61	[2, Section 3.4]	V	●	○	
USERAUTH_PASSWD_CHANGEREQ	USERAUTH_PASSWD_CHANGEREQ	60	[6, Section 8]	V	○	●	
USERAUTH_PK_OK	USERAUTH_PK_OK	60	[6, Section 7]	C	○	●	
USERAUTH_REQUEST_HOSTBASED	USERAUTH_REQUEST	50	[6, Section 9]	C	●	○	Method `hostbased`
USERAUTH_REQUEST_KEYBOARD_INTERACTIVE	USERAUTH_REQUEST	50	[2, Section 3.1]	V	●	○	Method `keyboard-interactive`
USERAUTH_REQUEST_NONE	USERAUTH_REQUEST	50	[6, Section 5.2]	V	●	○	Method `none`
USERAUTH_REQUEST_PASSWORD	USERAUTH_REQUEST	50	[6, Section 8]	V	●	○	Method `password`
USERAUTH_REQUEST_PUBLICKEY	USERAUTH_REQUEST	50	[6, Section 7]	C	●	○	Method `publickey`
USERAUTH_REQUEST_PUBLICKEY_HOSTBOUND_OPENSSH	USERAUTH_REQUEST	50	[10, Section 3.1]	C	●	○	Method `publickey-hostbound-v00@`
USERAUTH_REQUEST_UNKNOWN	USERAUTH_REQUEST	50	*n/a*	S	○	○	Method `unknown`
USERAUTH_SUCCESS	USERAUTH_SUCCESS	52	[6, Section 5.1]	S	○	●	

Connection protocol learning symbols used by the learner. For each symbol, the table lists the name of the symbol without the `MSG_` prefix, the specification name without the `SSH_MSG_` prefix, the message ID for the corresponding message type, a reference to the relevant section of the specification, the message class, whether the message is sent by client or server, and additional notes such as channel or request type of the message. A half-filled circle (◐) indicates that the specification discourages, but does not disallow, the message direction in question. The different message classes are described in Section 3.3 of the paper, referred to as *S* for static, *C* for cryptographic, and *V* for variable messages.

Symbol	Message Type	ID	Reference	Class	Send by		Note
					C	S	
CHANNEL_CLOSE	CHANNEL_CLOSE	97	[7, Section 5.3]	V	●	●	
CHANNEL_DATA	CHANNEL_DATA	94	[7, Section 5.2]	V	●	●	
CHANNEL_EOF	CHANNEL_EOF	96	[7, Section 5.3]	V	●	●	
CHANNEL_EXTENDED_DATA	CHANNEL_EXTENDED_DATA	95	[7, Section 5.2]	V	●	●	
CHANNEL_FAILURE	CHANNEL_FAILURE	100	[7, Section 5.4]	V	●	●	
CHANNEL_OPEN_CONFIRMATION	CHANNEL_OPEN_CONFIRMATION	91	[7, Section 5.4]	V	●	●	
CHANNEL_OPEN_DIRECT_STREAMLOCAL_OPENSSH	CHANNEL_OPEN	90	[10, Section 2.4]	V	●	○	direct-streamlocal@
CHANNEL_OPEN_DIRECT_TCPIP	CHANNEL_OPEN	90	[7, Section 7.2]	V	●	◐	direct-tcpip
CHANNEL_OPEN_FAILURE	CHANNEL_OPEN_FAILURE	92	[7, Section 5.1]	V	●	●	
CHANNEL_OPEN_FORWARDED_STREAMLOCAL_OPENSSH	CHANNEL_OPEN	90	[10, Section 2.4]	V	○	●	forwarded-streamlocal@
CHANNEL_OPEN_FORWARDED_TCPIP	CHANNEL_OPEN	90	[7, Section 7.2]	V	●	●	forwarded-tcpip
CHANNEL_OPEN_SESSION	CHANNEL_OPEN	90	[7, Section 6.1]	V	●	◐	session
CHANNEL_OPEN_TUN_OPENSSH	CHANNEL_OPEN	90	[10, Section 2.3]	V	●	○	tun@
CHANNEL_OPEN_UNKNOWN	CHANNEL_OPEN	90	*n/a*	V	○	○	unknown
CHANNEL_OPEN_X11	CHANNEL_OPEN	90	[7, Section 6.3.2]	V	●	●	x11
CHANNEL_REQUEST_AUTH_AGENT_OPENSSH	CHANNEL_REQUEST	98	[9, Section 4.2]	V	●	○	auth-agent-req@
CHANNEL_REQUEST_BREAK	CHANNEL_REQUEST	98	[11, Section 3]	V	●	○	break
CHANNEL_REQUEST_ENV	CHANNEL_REQUEST	98	[7, Section 6.4]	V	●	◐	env
CHANNEL_REQUEST_EOW_OPENSSH	CHANNEL_REQUEST	98	[10, Section 2.1]	V	●	●	eow@
CHANNEL_REQUEST_EXEC	CHANNEL_REQUEST	98	[7, Section 6.5]	V	●	◐	exec
CHANNEL_REQUEST_EXIT_SIGNAL	CHANNEL_REQUEST	98	[7, Section 6.10]	V	◐	●	exit-signal
CHANNEL_REQUEST_EXIT_STATUS	CHANNEL_REQUEST	98	[7, Section 6.10]	V	◐	●	exit-status
CHANNEL_REQUEST_PTY_REQ	CHANNEL_REQUEST	98	[7, Section 6.2]	V	●	◐	pty-req
CHANNEL_REQUEST_SHELL	CHANNEL_REQUEST	98	[7, Section 6.5]	V	●	◐	shell
CHANNEL_REQUEST_SIGNAL	CHANNEL_REQUEST	98	[7, Section 6.9]	V	●	◐	signal
CHANNEL_REQUEST_SUBSYSTEM	CHANNEL_REQUEST	98	[7, Section 6.5]	V	●	◐	subsystem
CHANNEL_REQUEST_UNKNOWN	CHANNEL_REQUEST	98	*n/a*	V	○	○	unknown
CHANNEL_REQUEST_WINDOW_CHANGE	CHANNEL_REQUEST	98	[7, Section 6.7]	V	●	○	window-change
CHANNEL_REQUEST_X11_REQ	CHANNEL_REQUEST	98	[7, Section 6.3.1]	V	●	●	x11-req
CHANNEL_REQUEST_XON_XOFF	CHANNEL_REQUEST	98	[7, Section 6.8]	V	○	●	xon-xoff
CHANNEL_SUCCESS	CHANNEL_SUCCESS	99	[7, Section 5.4]	V	●	●	
CHANNEL_WINDOW_ADJUST	CHANNEL_WINDOW_ADJUST	93	[7, Section 5.2]	V	●	●	
GLOBAL_REQUEST_CANCEL_STREAMLOCAL_FORWARD_OPENSSH	GLOBAL_REQUEST	80	[10, Section 2.4]	V	●	○	cancel-streamlocal-forward@
GLOBAL_REQUEST_CANCEL_TCPIP_FORWARD	GLOBAL_REQUEST	80	[7, Section 7.1]	V	●	◐	cancel-tcpip-forwarded
GLOBAL_REQUEST_HOSTKEYS_OPENSSH	GLOBAL_REQUEST	80	[10, Section 2.5]	C	○	●	hostkeys-00@
GLOBAL_REQUEST_HOSTKEYS_PROVE_OPENSSH	GLOBAL_REQUEST	80	[10, Section 2.5]	C	●	○	hostkeys-prove-00@
GLOBAL_REQUEST_NO_MORE_SESSIONS_OPENSSH	GLOBAL_REQUEST	80	[10, Section 2.2]	V	●	○	no-more-sessions@
GLOBAL_REQUEST_STREAMLOCAL_FORWARD_OPENSSH	GLOBAL_REQUEST	80	[10, Section 2.4]	V	●	○	streamlocal-forward@
GLOBAL_REQUEST_TCPIP_FORWARD	GLOBAL_REQUEST	80	[7, Section 7.1]	V	●	◐	tcpip-forward
GLOBAL_REQUEST_UNKNOWN	GLOBAL_REQUEST	80	*n/a*	S	○	○	unknown
REQUEST_FAILURE	REQUEST_FAILURE	82	[7, Section 4]	S	●	●	
REQUEST_SUCCESS	REQUEST_SUCCESS	81	[7, Section 4]	S	●	●	
UNKNOWN_ID_CHANNEL_RELATED	*n/a*	101	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_CONNECTION_GENERIC	*n/a*	83	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_RESERVED_CLIENT	*n/a*	128	*n/a*	S	○	○	Message ID only
UNKNOWN_ID_RESERVED_PRIVATE	*n/a*	192	*n/a*	S	○	○	Message ID only

References

[1] Denis Bider. 2018. Extension Negotiation in the Secure Shell (SSH) Protocol. RFC 8308. doi:10.17487/RFC8308

[2] Martin Forssen and Frank Cusack. 2006. Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). RFC 4256. doi:10.17487/RFC4256

[3] Markus Friedl, Niels Provos, and William A. Simpson. 2006. Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4419. doi:10.17487/RFC4419

[4] Ben Harris. 2006. RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4432. doi:10.17487/RFC4432

[5] Panos Kampanakis, Douglas Stebila, and Torben Hansen. 2025. *PQ/T Hybrid Key Exchange in SSH*. Internet-Draft draft-ietf-sshm-mlkem-hybrid-kex-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-sshm-mlkem-hybrid-kex/00/ Work in Progress.

[6] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Authentication Protocol. RFC 4252. doi:10.17487/RFC4252

[7] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Connection Protocol. RFC 4254. doi:10.17487/RFC4254

[8] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253. doi:10.17487/RFC4253

[9] Damien Miller. 2025. *SSH Agent Protocol*. Internet-Draft draft-ietf-sshm-ssh-agent-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-sshm-ssh-agent/02/ Work in Progress.

[10] Damien Miller, Markus Friedl, Mike Frysinger, Todd C. Miller, and Darren Tucker. 2024. This documents OpenSSH's deviations and extensions to the published SSH protocol. https://cvsweb.openbsd.org/cgi-bin/cvsweb/src/usr.bin/ssh/PROTOCOL?rev=1.55 Accessed: 2025-04-14.

[11] Phillip Remaker and Joseph Galbraith. 2006. The Secure Shell (SSH) Session Channel Break Extension. RFC 4335. doi:10.17487/RFC4335

[12] Douglas Stebila and Jonathan Green. 2009. Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer. RFC 5656. doi:10.17487/RFC5656