Table 1: Resulting algorithm choices for state learning with each implementation based on the algorithm lists. Key exchange algorithms are configured separately and therefore not included in this table. The choice of algorithms is used symmetrically for both connection directions. An *<implicit>* MAC algorithm denotes that an AEAD encryption algorithm is used and no separate MAC algorithm is negotiated.

Implementation	Host Key	Encryption	MAC	Compression
AsyncSSH	`rsa-sha2-512`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
Bitvise SSH	`ecdsa-sha2-nistp384`	`aes128-ctr`	`hmac-sha2-256`	none
Dropbear	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
Erlang SSH	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
Golang x/crypto/ssh	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
LANCOM LCOS	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
libssh	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
OpenSSH	`ecdsa-sha2-nistp256`	`chacha20-poly1305@openssh.com`	*<implicit>*	none
Tectia SSH	`rsa-sha2-512`	`aes128-ctr`	`hmac-sha2-256`	none
TinySSH	`ssh-ed25519`	`chacha20-poly1305@openssh.com`	*<implicit>*	none

We distinguish different protocol flows inside the transport layer protocol based on the key exchange flow type, assuming that different algorithms of the same flow type do not influence the resulting state machine. We, therefore, select one algorithm of each flow type for evaluation. The mapping between flow type and key exchange algorithm is listed below. For TinySSH, we use `curve25519-sha256` as the ECDH algorithm because of the lack of support for the NIST P-256 curve. For all evaluation runs, the key exchange algorithm list consists of the key exchange algorithm and the strict kex identifier.

- DH: `diffie-hellman-group14-sha256`
- DHGEX: `diffie-hellman-group-exchange-sha256`
- ECDH: `ecdh-sha2-nistp256`
- RSA: `rsa2048-sha256`
- PQ-Hybrid: `sntrup761x25519-sha512@openssh.com`

We use reasonable defaults for all other algorithm lists based on OpenSSH 9.9p2. Compression is disabled (none). The algorithm lists are given in order of preference below. The SSH protocol chooses the first algorithm from the client's list, which is also supported by the server. This results in the algorithm choices listed in Table 1 for the evaluated server implementations.

Host key algorithms.

- `ecdsa-sha2-nistp256-cert-v01@openssh.com`
- `ecdsa-sha2-nistp384-cert-v01@openssh.com`
- `ecdsa-sha2-nistp521-cert-v01@openssh.com`
- `ssh-ed25519-cert-v01@openssh.com`
- `rsa-sha2-512-cert-v01@openssh.com`
- `rsa-sha2-256-cert-v01@openssh.com`
- `ssh-rsa-cert-v01@openssh.com`
- `ecdsa-sha2-nistp256`
- `ecdsa-sha2-nistp384`
- `ecdsa-sha2-nistp521`
- `ssh-ed25519`
- `rsa-sha2-512`
- `rsa-sha2-256`
- `ssh-rsa`

Encryption algorithms.

- `chacha20-poly1305@openssh.com`
- `aes128-ctr`
- `aes192-ctr`
- `aes256-ctr`
- `aes128-gcm@openssh.com`
- `aes256-gcm@openssh.com`

MAC algorithms.

- `umac-64-etm@openssh.com`
- `umac-128-etm@openssh.com`
- `hmac-sha2-256-etm@openssh.com`
- `hmac-sha2-512-etm@openssh.com`
- `hmac-sha1-etm@openssh.com`
- `umac-64@openssh.com`
- `umac-128@openssh.com`
- `hmac-sha2-256`
- `hmac-sha2-512`
- `hmac-sha1`