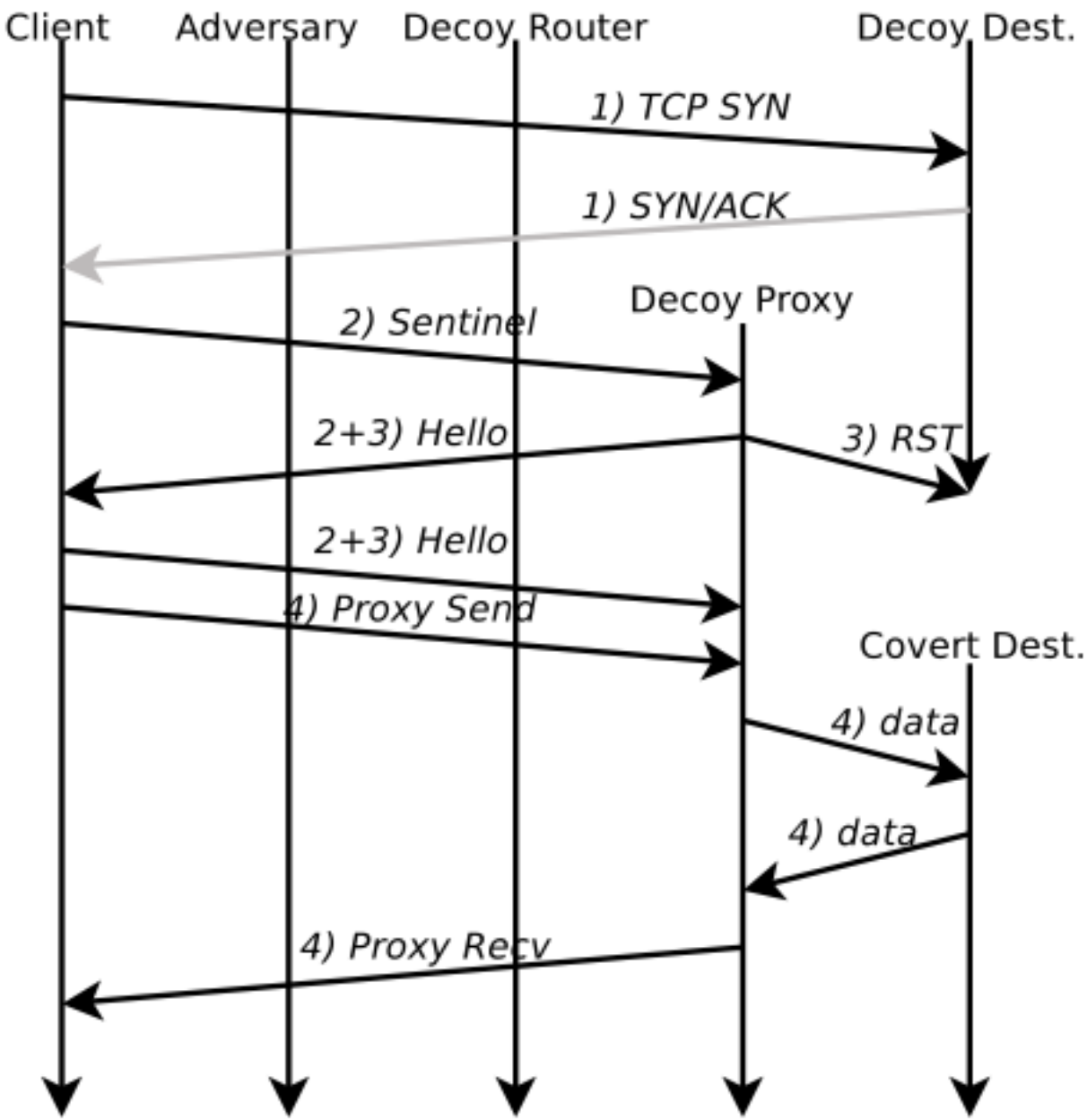
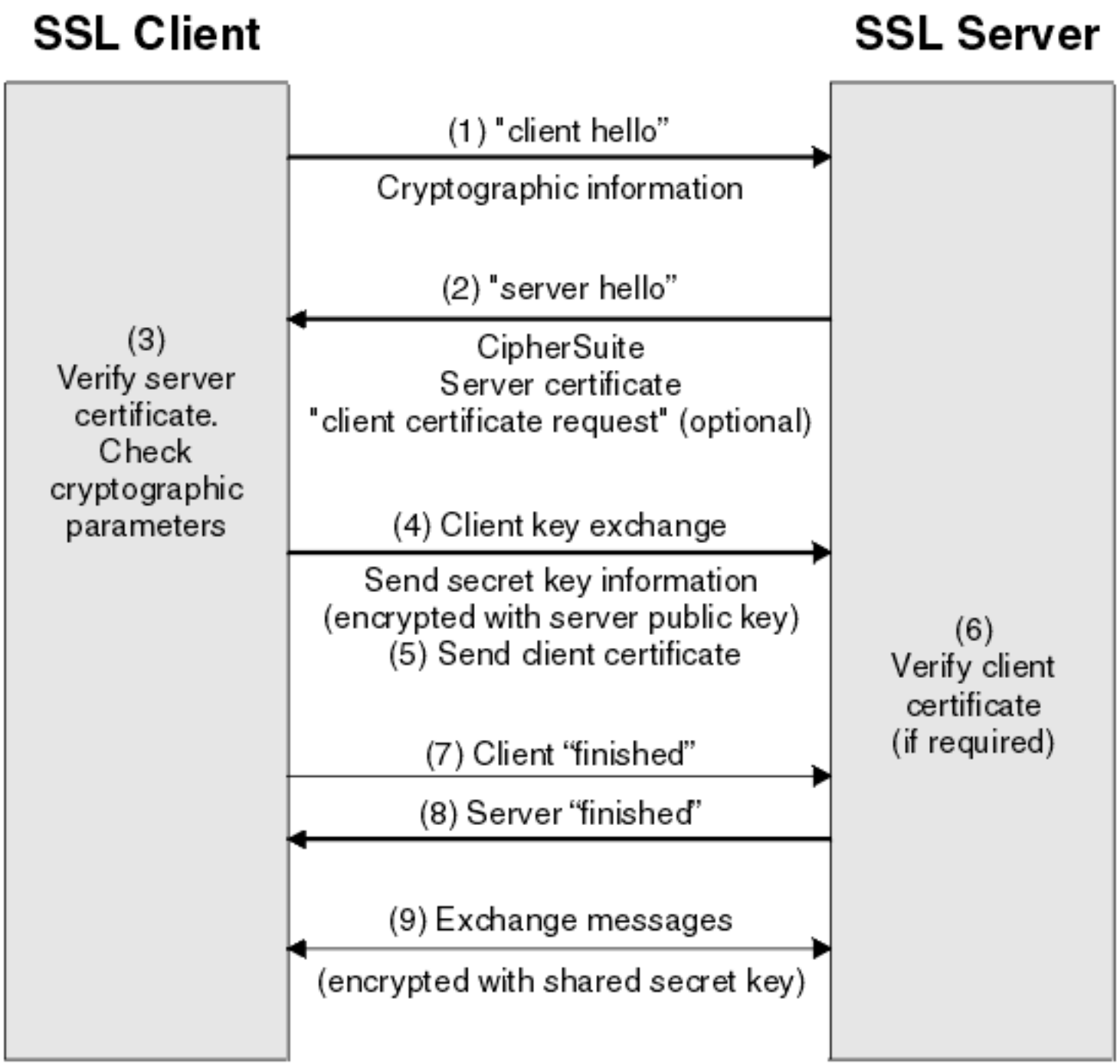

Client
Adversary
Decoy Router
Decoy Dest.
1) TCP SYN
1) SYN/ACK
Decoy Proxy
2) Sentinel
2+3) Hello
3) RST
2+3) Hello
4) Proxy Send
Covert Dest.
4) data
4) data
4) Proxy Recv


SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3)
Verify server certificate.
Check cryptographic parameters
(4) Client key exchange
Send secret key information
(encrypted with server public key)
(5) Send client certificate
(6)
Verify client certificate
(if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Client Hello. The client initiates a session by sending a Client Hello message to the server. The Client Hello message contains:

- **Version Number.** The client sends the version number corresponding to the highest version it supports. Version 2 is used for SSL 2.0, version 3 for SSL 3.0, and version 3.1 for TLS. Although the IETF RFC for TLS is TLS version 1.0, the protocol uses 3.1 in the version field to indicate that it is a higher level (newer and with more functionality) than SSL 3.0.

- **Randomly Generated Data.** ClientRandom[32], the random value, is a 4-byte number that consists of the client's date and time plus a 28-byte randomly generated number that will ultimately be used with the server random value to generate a master secret from which the encryption keys will be derived.

- **Session Identification (if any).** The sessionID is included to enable the client to resume a previous session. Resuming a previous session can be useful, because creating a new session requires processor-intensive public key operations that can be avoided by resuming an existing session with its established session keys. Previous session information, identified by the sessionID, is stored in the respective client and server session caches.

- **Cipher Suite.** The A list of cipher suites available on the client. An example of a cipher suite is TLS_RSA_WITH_DES_CBC_SHA, where TLS is the protocol version, RSA is the algorithm that will be used for the key exchange, DES_CBC is the encryption algorithm (using a 56-bit key in CBC mode), and SHA is the hash function.

- **Compression Algorithm.** The requested compression algorithm (none currently supported).

It is also worth noting that the flow table can be several orders-of-magnitude smaller than the forwarding table in an equivalent Ethernet switch. In an Ethernet switch, the table is sized to minimize broadcast traffic: as switches flood during learning, this can swamp links and makes the network less secure.[5] As a result, an Ethernet switch needs to remember all the addresses it's likely to encounter; even small wiring closet switches typically contain a million entries. Ethane Switches, on the other hand, can have much smaller

two-way hashing scheme [9]. A typical commercial enterprise Ethernet switch today holds 1 million Ethernet addresses (6MB, but larger if hashing is used), 1 million IP addresses (4MB of TCAM),

Table 1. Scalability Table

Name	WS-SUP720-3B	WS-SUP720-3BXL	VS-S720-10G-3C *	VS-S720-10G-3CXL*
MAC Entries	64,000	64,000	96,000	96,000
Routes	256,000 (IPv4); 128,000 (IPv6)	1,000,000 (IPv4); 500,000 (IPv6)	256,000 (IPv4); 128,000 (IPv6)	1,000,000 (IPv4); 500,000 (IPv6)

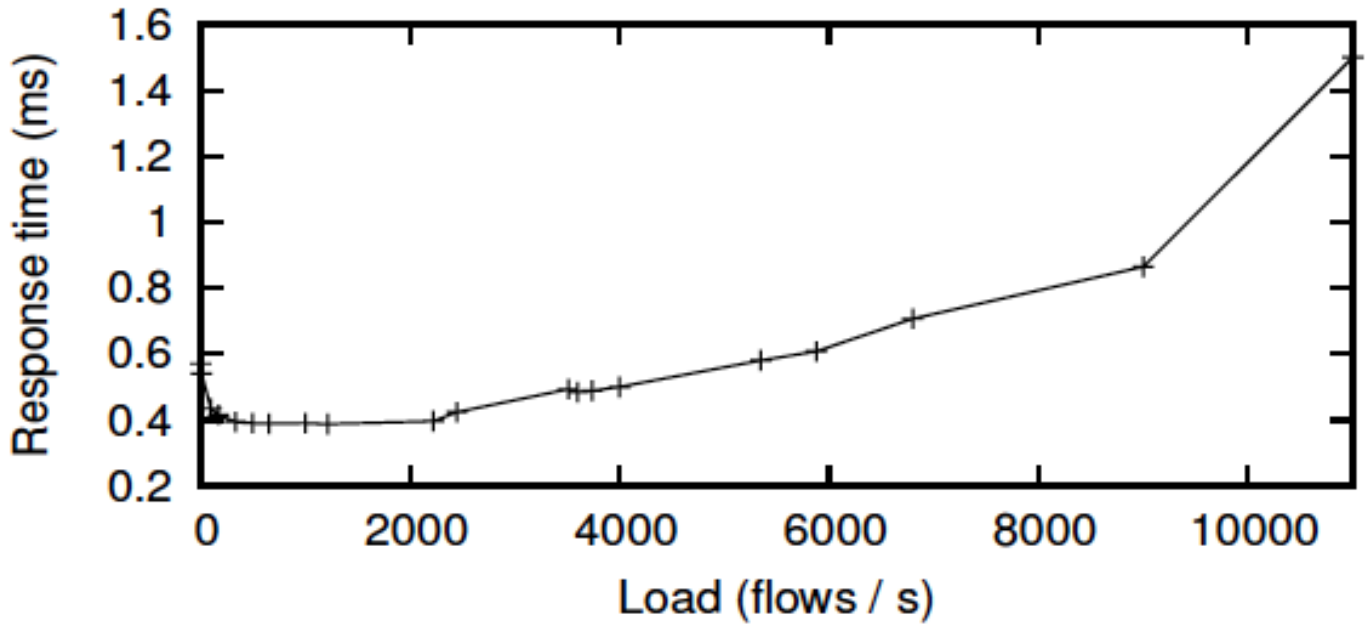


Figure 6: Flow-setup times as a function of Controller load. Packet sizes were 64B, 128B and 256B, evenly distributed.

heads. The Controller was configured with a policy file of 50 rules and 100 registered principles; routes were precalculated and cached. Under these conditions, the system could handle 650,845 bind events per second and 16,972,600 permission checks per second. The

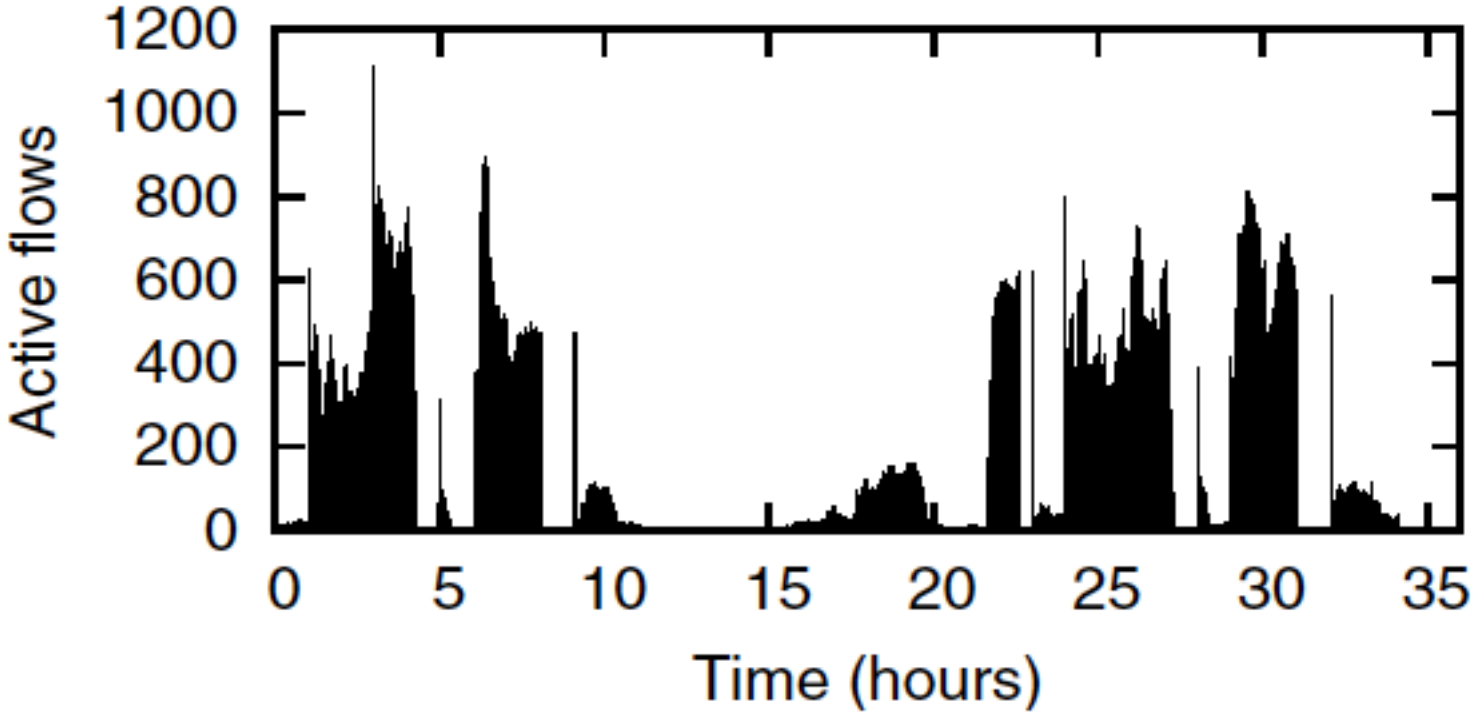


Figure 7: Active flows for LBL network [19].