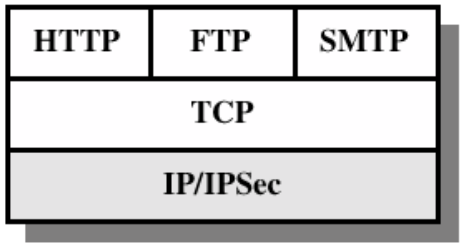
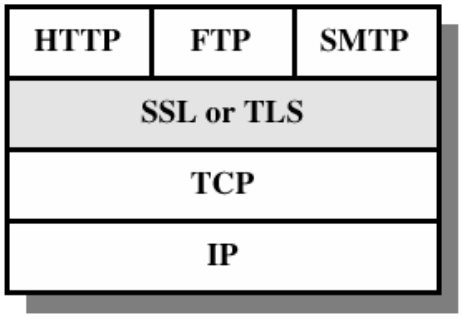
Secure Socket Layer

Mridul Sankar Barik
Dept. of Comp. sc. & Engg.
Jadavpur University

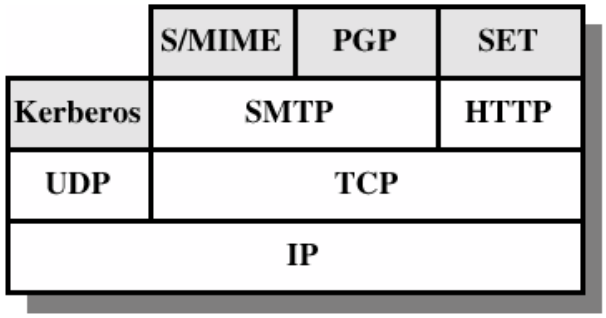
Security Facilities in TCP/IP Protocol Stack



(a) Network Level

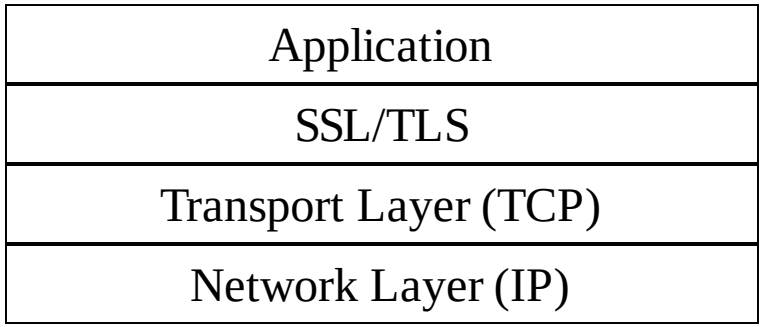


(b) Transport Level



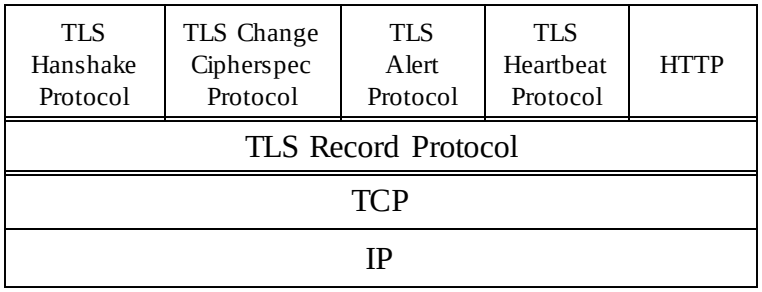
(c) Application Level

Location of SSL



- SSL is build on top of TCP
- Provides a TCP like interface
- Provides end-to-end security services for applications
  - Entity authentication
  - Message integrity
  - Confidentiality
- Netscape developed SSL in 1994
  - Released version 2 and 3 in 1995
- TLS is IETF version of SSL

TLS Architecture



- TLS Record Protocol
  - Provides basic security services to various higher layer protocols
- TLS Handshake, TLS Change Cipher Spec, TLS Alert, TLS Heartbeat protocol
  - Protocols used in management of SSL exchanges

TLS Services

- Fragmentation
  - SSL divides the data into blocks of 2^{14} bytes or less
- Compression
  - Optional
- Message Integrity
  - SSL uses keyed hash function to compute MAC
- Confidentiality
  - Original data and MAC are encrypted using symmetric key cryptography
- Framing
  - SSL header is added to the encrypted payload

TLS Architecture

- TLS session
  - An association between client & server
  - Created by the Handshake Protocol
  - Define a set of information shared by two parties
    - Session identifier
    - Certificate authenticating each party
    - Compression method
    - Cipher suite
    - Master secret that is used to create other keys
  - May be shared by multiple TLS connections
- TLS connection
  - A peer-to-peer, communications link
  - Each connection is associated with one TLS session

Session Parameters

- Session identifier:
  - Arbitrary byte sequence chosen by the server
- Peer certificate:
  - An X509.v3 certificate of the peer
- Compression method:
  - Compression algorithm used prior encryption (optional)
- Cipher spec:
  - Specifies a data encryption algorithm (null, DES, etc.), a hash algorithm (MD5 or SHA-1) and hash size
- Master secret:
  - 48-byte secret shared between client and server
- Is resumable:
  - Whether the session can be used to initiate new connection

Connection Parameters

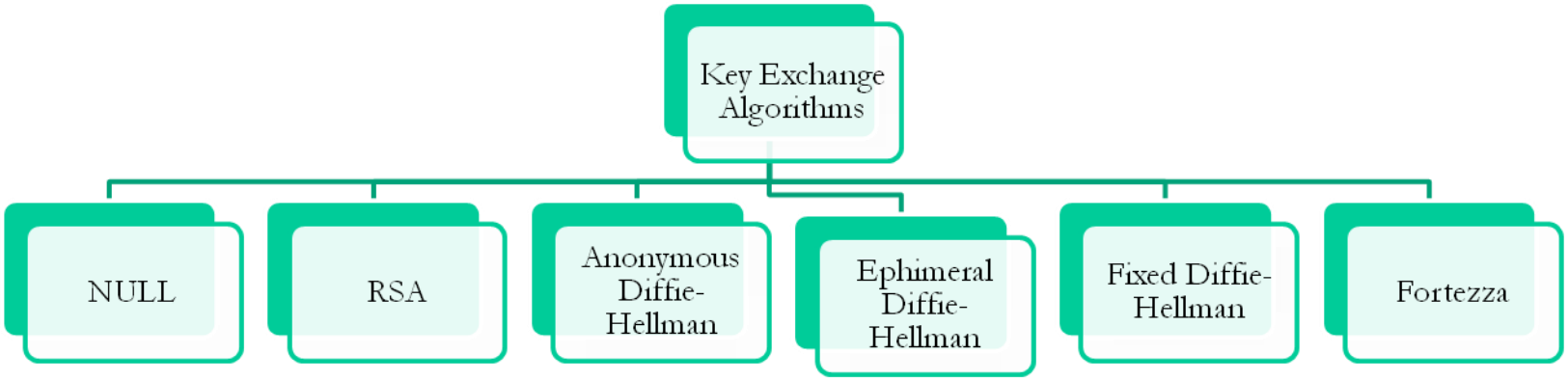
- Server and client random:
  - Byte sequences chosen by the server and client
- Server write MAC secret:
  - Secret key used in MAC operations on data sent by the server
- Client write MAC secret:
  - Secret key used in MAC operations on data sent by the client
- Server write key:
  - Conventional encryption key for data encrypted by server and decrypted by client
- Client write key:
  - Conventional encryption key for data encrypted by client and decrypted by server

Connection Parameters

- Initialization vectors:
  - Required for each key when a block cipher in CBC mode is used
  - Initialized by TLS Handshake protocol
  - Final cipher block from each record is used as IV with the following record
- Sequence numbers:
  - Each party maintains separate sequence numbers for transmitted and received messages for each connection
  - When a party sends or receives change cipher spec message, appropriate sequence number is set to zero
  - Sequence numbers may not exceed $2^{64}-1$

Key Exchange Algorithms

- TLS requires six cryptographic secrets
  - Four keys and two initialization vectors
- To create these secrets one pre master secret must be established
- TLS defines six key-exchange methods to establish this pre-master secret



Key Exchange Algorithms

- NULL
  - No pre-master secret is established between the client and the server
- RSA
  - Client generates random 48-bytes pre-master secret and encrypts it with server's RSA public key
- Anonymous Diffie-Hellman
  - Simplest most insecure method
  - Diffie-Hellman half keys are sent in plain text
  - Neither party is known to each other
  - Prone to man-in-the-middle attack

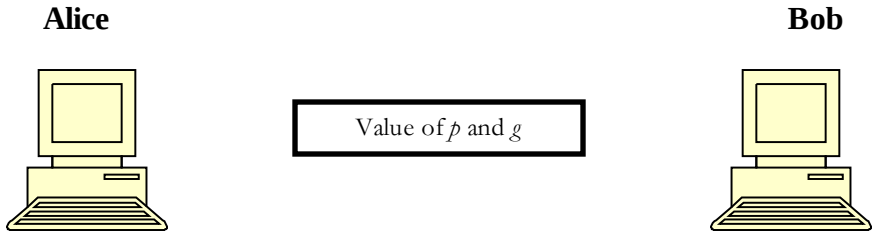
Diffie-Hellman Key Exchange

- First public-key type scheme proposed by Diffie & Hellman in 1976
- Is a practical method for public exchange of a secret key
- Used in a number of commercial products

Diffie-Hellman Key Exchange

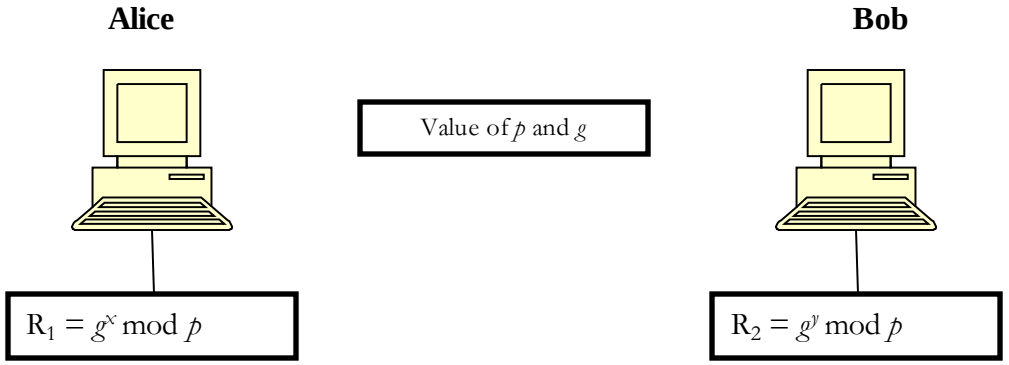
- Primitive root of a prime number p is one whose powers generate all integers from 1 to p-1
  - If a is a primitive root of the prime number p, then the numbers $a \bmod p$, $a^2 \bmod p$, $a^3 \bmod p$, …, $a^{p-1} \bmod p$ are distinct and consists of integers from 1 through p-1 in some permutation
- For any integer b and primitive root a of prime number p one can find a unique integer i such that $b = a^i \bmod p$ where $0 < i \leq (p-1)$
  - Exponent i is referred to as the discrete logarithm or index of b for the base a mod p , denoted as $\text{ind}_{a,\,p}(b)$
  - Very difficult to calculate discrete logarithm for large primes (infeasible)

Diffie-Hellman Key Exchange



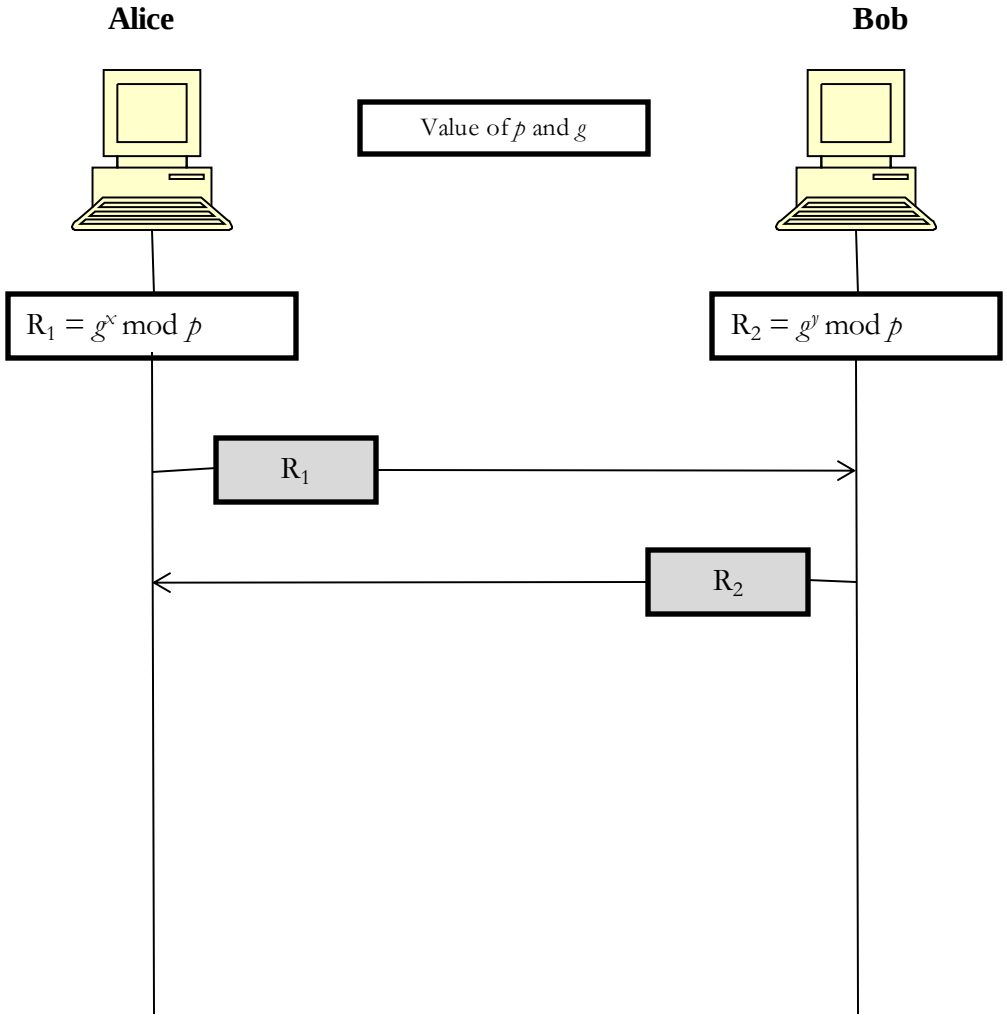
- Two parties chooses two numbers p and g
  - p is a large prime number of the order 300 decimal digits (1024 bits)
  - g is generator of the order p-1
  - Need not be confidential

Diffie-Hellman Key Exchange



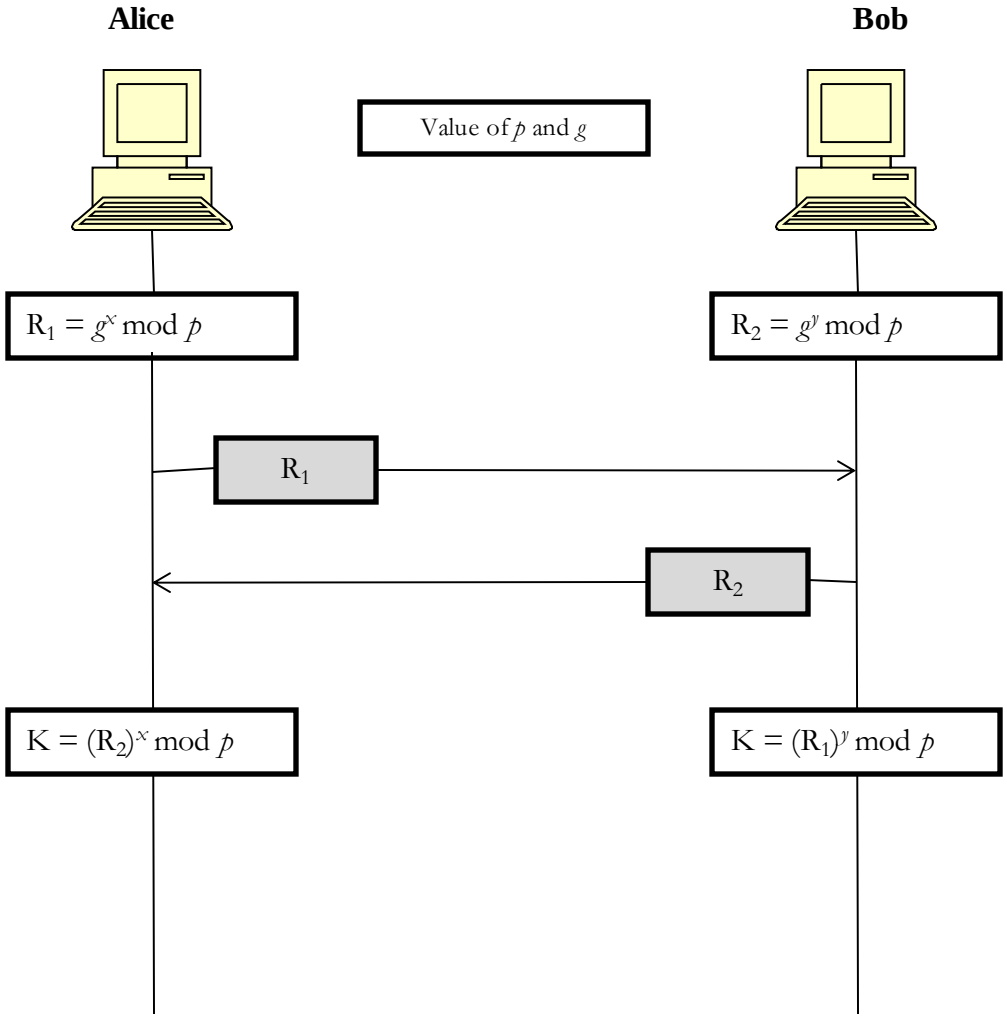
- Alice chooses a large random number x such that $0 \leq x \leq p-1$ and calculates $R_1 = g^x \bmod p$
- Bob chooses another large random number y such that $0 \leq y \leq p-1$ and calculates $R_2 = g^y \bmod p$

Diffie-Hellman Key Exchange



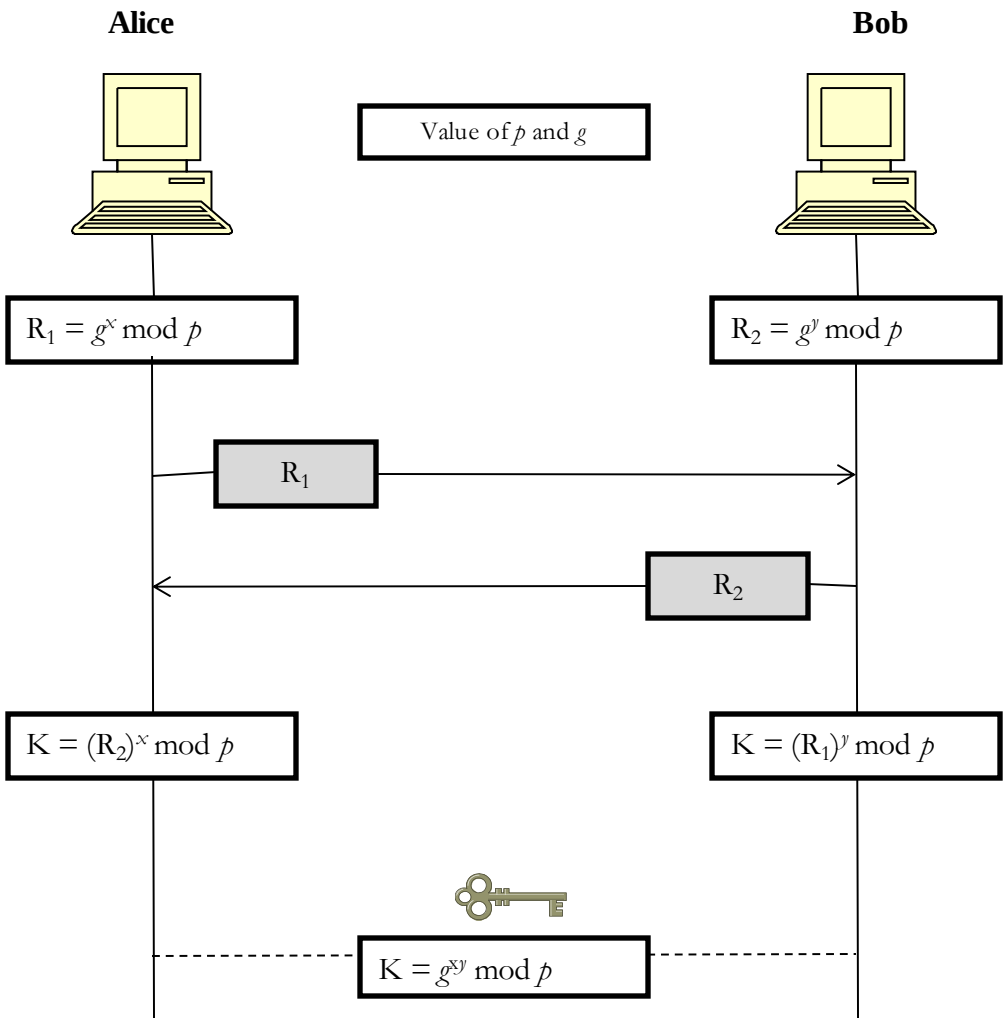
- Alice sends R_1 to Bob
- Bob sends R_2 to Alice

Diffie-Hellman Key Exchange



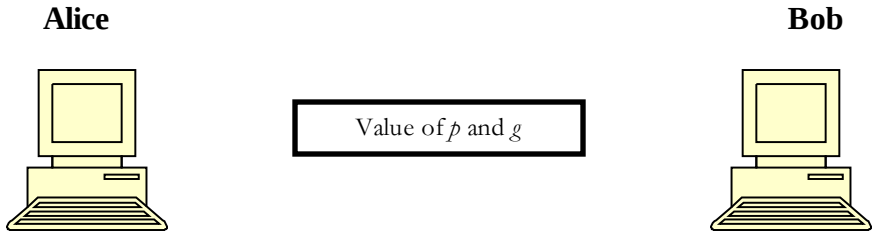
- Alice calculates $K = (R_2)^x \bmod p$
- Bob calculates $K = (R_1)^y \bmod p$

Diffie-Hellman Key Exchange



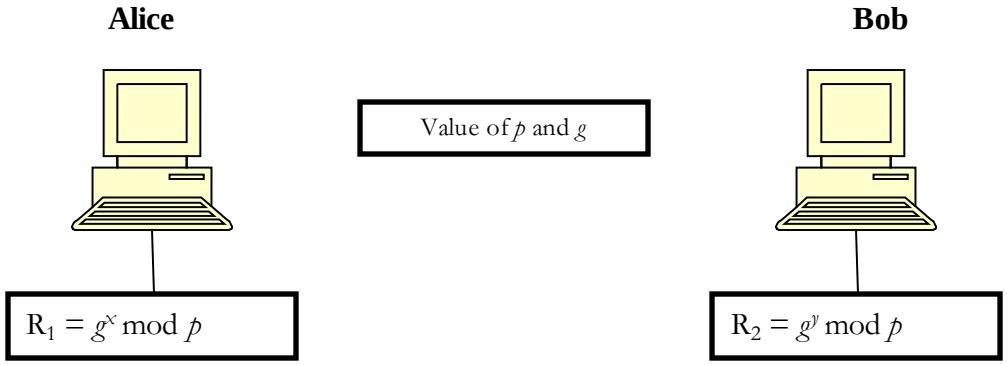
- K is the symmetric key for the session
- $K = (g^y \bmod p)^x \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p$

Diffie-Hellman Key Exchange - Example



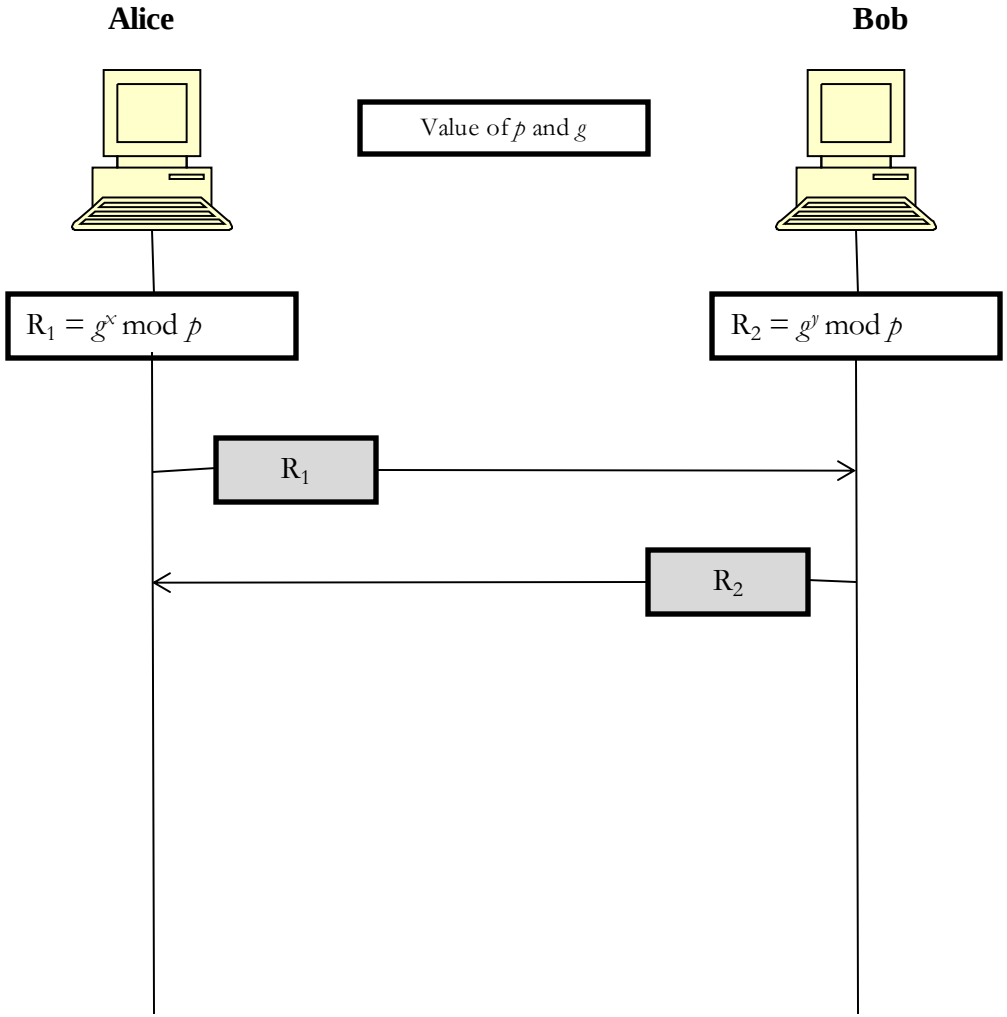
- Assume $g = 3$ and $p = 7$

Diffie-Hellman Key Exchange - Example



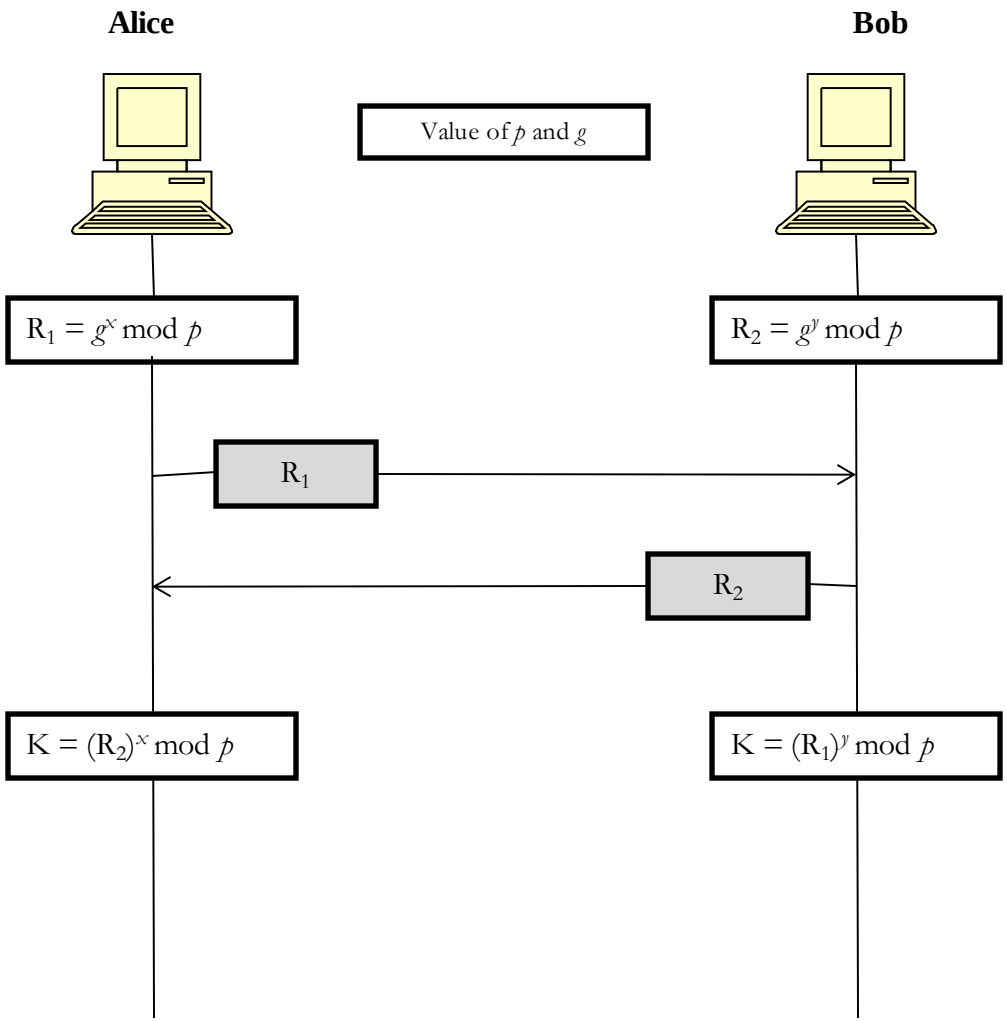
- Assume $g = 3$ and $p = 7$
- Alice chooses x=2 and calculates $R_1 = g^x \bmod p = 3^2 \bmod 7 = 2$
- Bob chooses y=5 and calculates $R_2 = g^y \bmod p = 3^5 \bmod 7 = 5$

Diffie-Hellman Key Exchange - Example



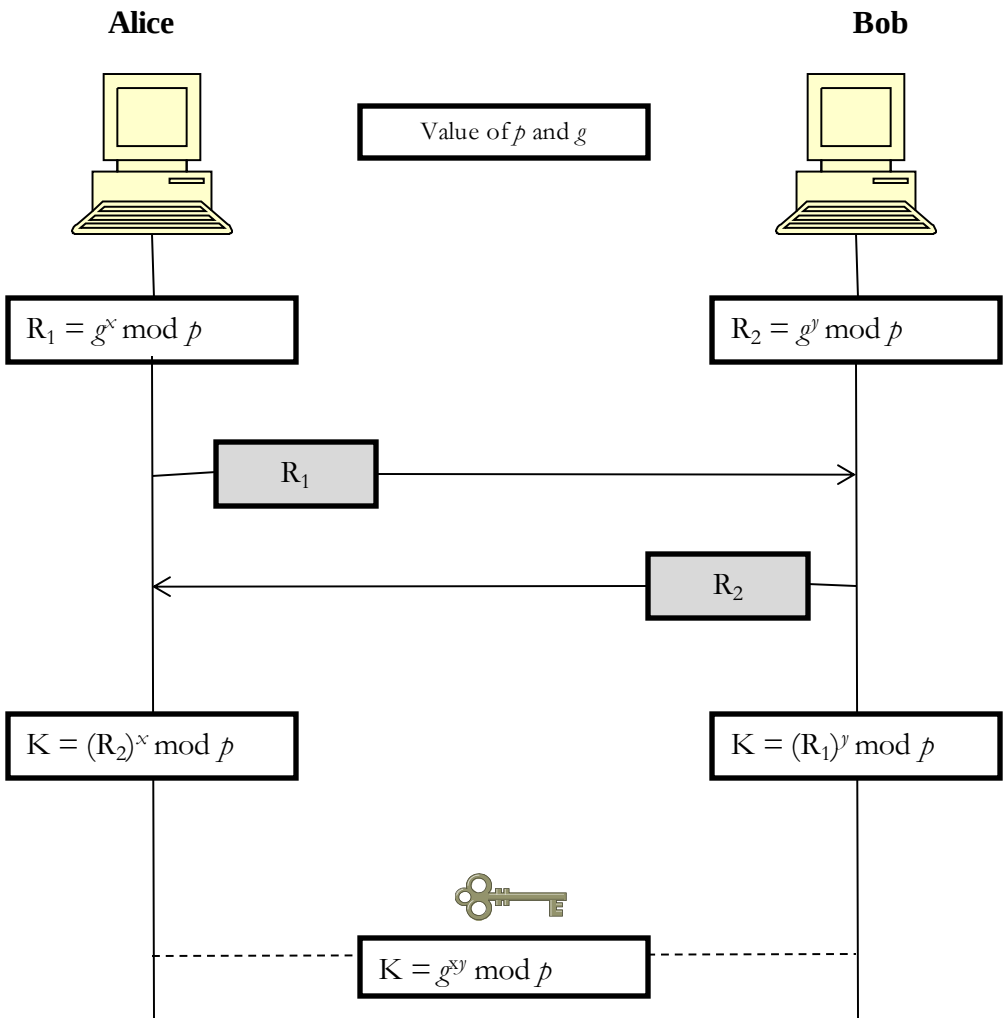
- Alice sends 2 to Bob
- Bob sends 5 to Alice

Diffie-Hellman Key Exchange - Example



- Assume $g = 3$ and $p = 7$
- Alice calculates $K = (R_2)^x \bmod p = (5)^2 \bmod 7 = 4$
- Bob calculates $K = (R_1)^y \bmod p = (2)^5 \bmod 7 = 4$

Diffie-Hellman Key Exchange - Example

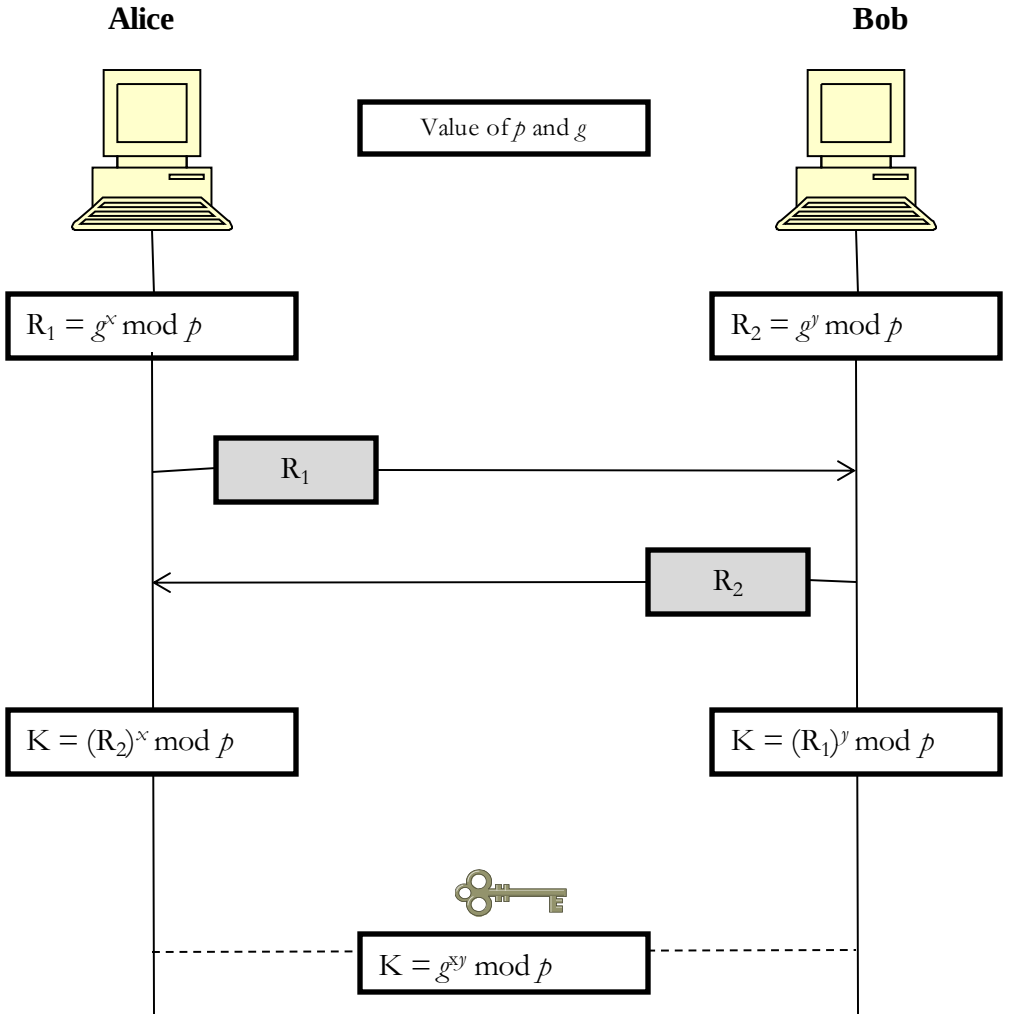


- The value of K is the same for both Alice and Bob
  - $g^{xy} \bmod p = 3^{10} \bmod 7 = 59049 \bmod 7$

Security of Diffie-Hellman Key Exchange

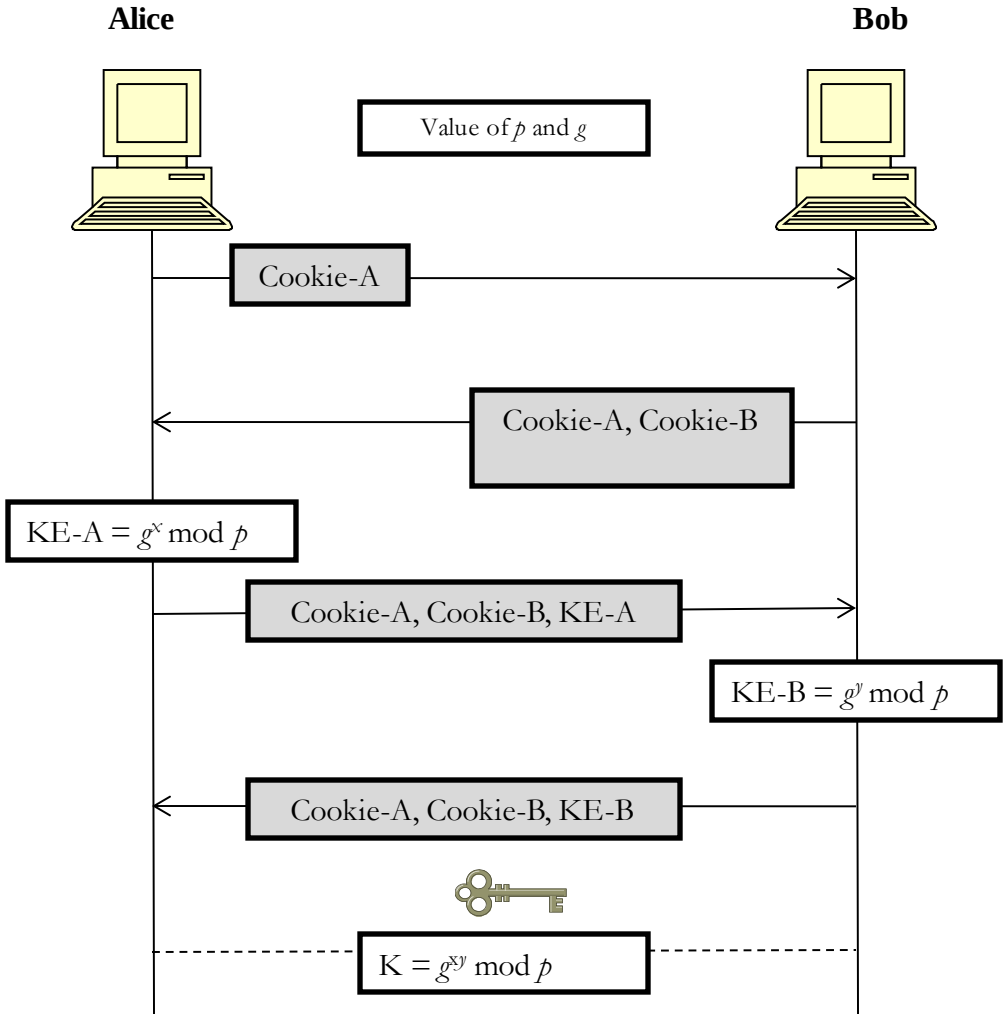
- Discrete Logarithm Attack
  - Security is based on difficulty of the discrete logarithm problem
  - Eve can intercept R_1 and R_2
    - Finds x from $R_1 = g^x \bmod p$
    - Finds y from $R_2 = g^y \bmod p$
    - Calculates key $K = g^{xy} \bmod p$

Security of Diffie-Hellman Key Exchange



- Clogging Attack or Denial of Service Attack
  - A malicious intruder can send many half-key ($g^x \bmod p$) messages to Bob, pretending that they are from different sources
  - Bob needs to calculate different responses ($g^y \bmod p$) and calculate full-key ($g^{xy} \bmod p$)
  - This keeps Bob busy may cause DoS

Security of Diffie-Hellman Key Exchange

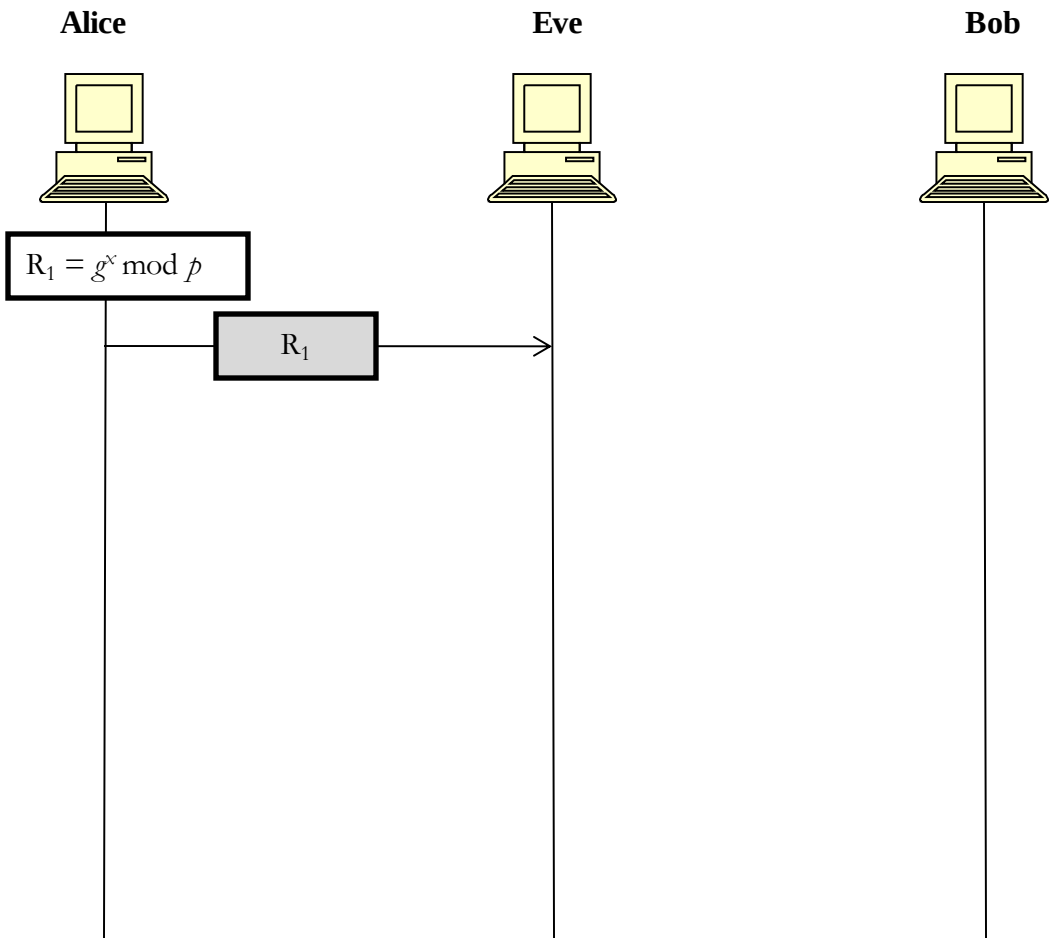


- Clogging Attack or Denial of Service Attack
  - Two parties use cookies

Security of Diffie-Hellman Key Exchange

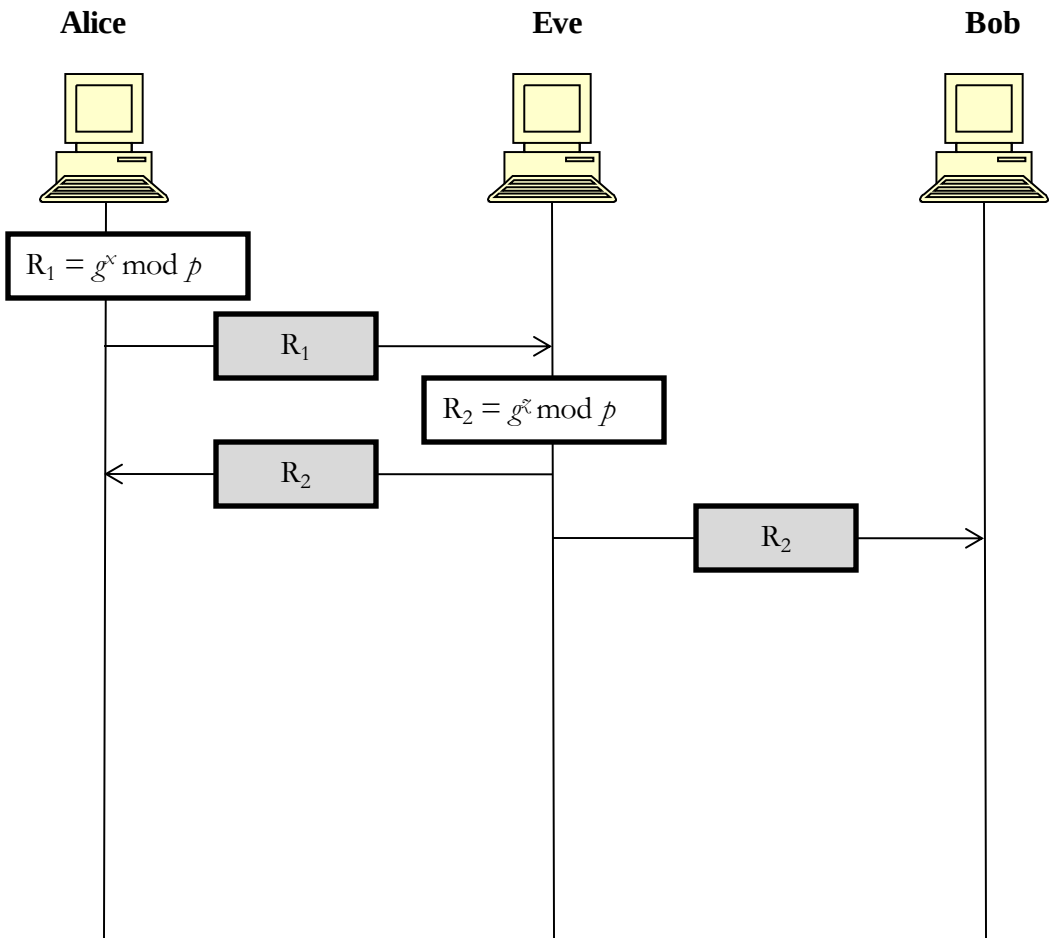
- Man in the Middle Attack
  - Eve can fool Alice and Bob by creating two keys
    - One between herself and Alice
    - Another between herself and Bob

Security of Diffie-Hellman Key Exchange



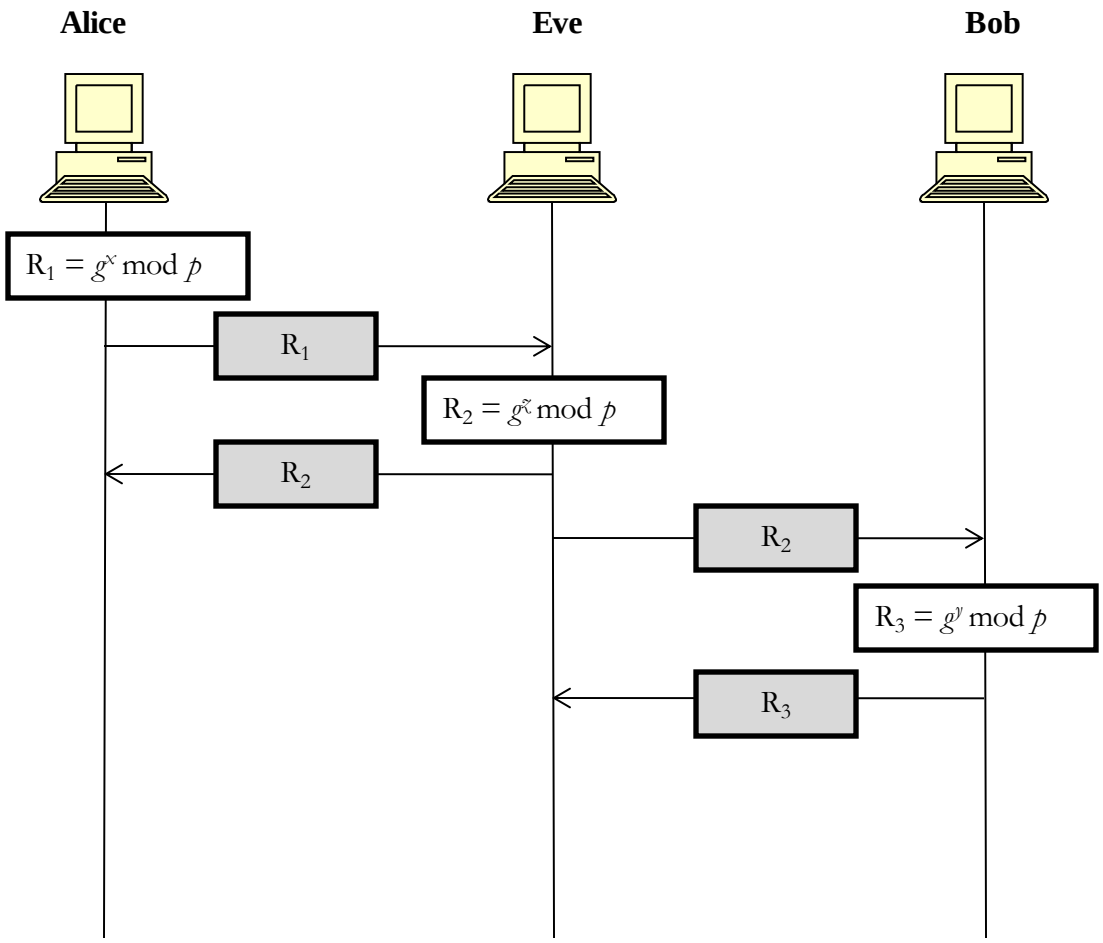
- Man in the Middle Attack
  - Alice chooses x, calculates $R_1 = g^x \bmod p$ and sends R_1 to Bob

Security of Diffie-Hellman Key Exchange



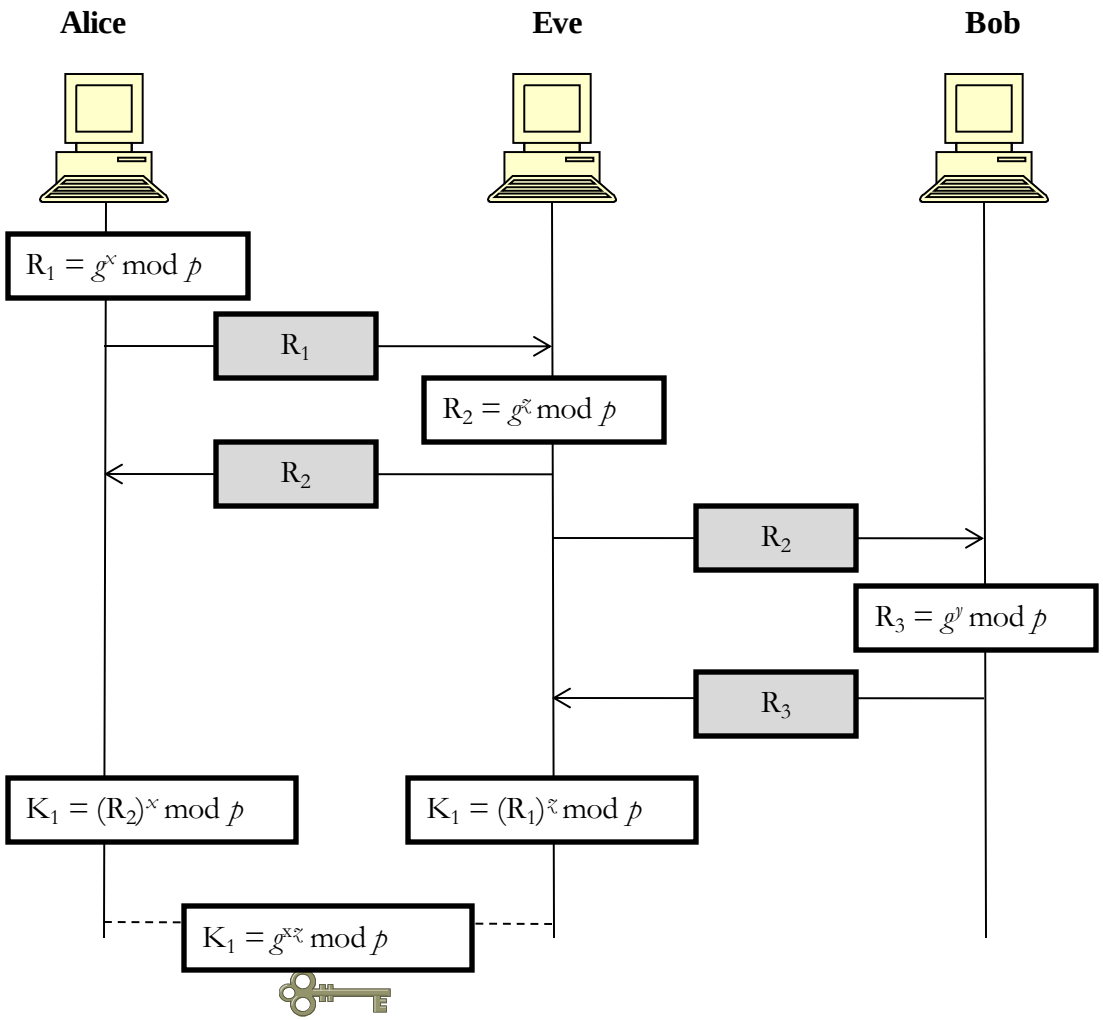
- Man in the Middle Attack
  - Eve, the intruder, intercepts R_1, chooses z, calculates $R_2 = g^z \bmod p$ and sends R_2 to both Alice and Bob

Security of Diffie-Hellman Key Exchange



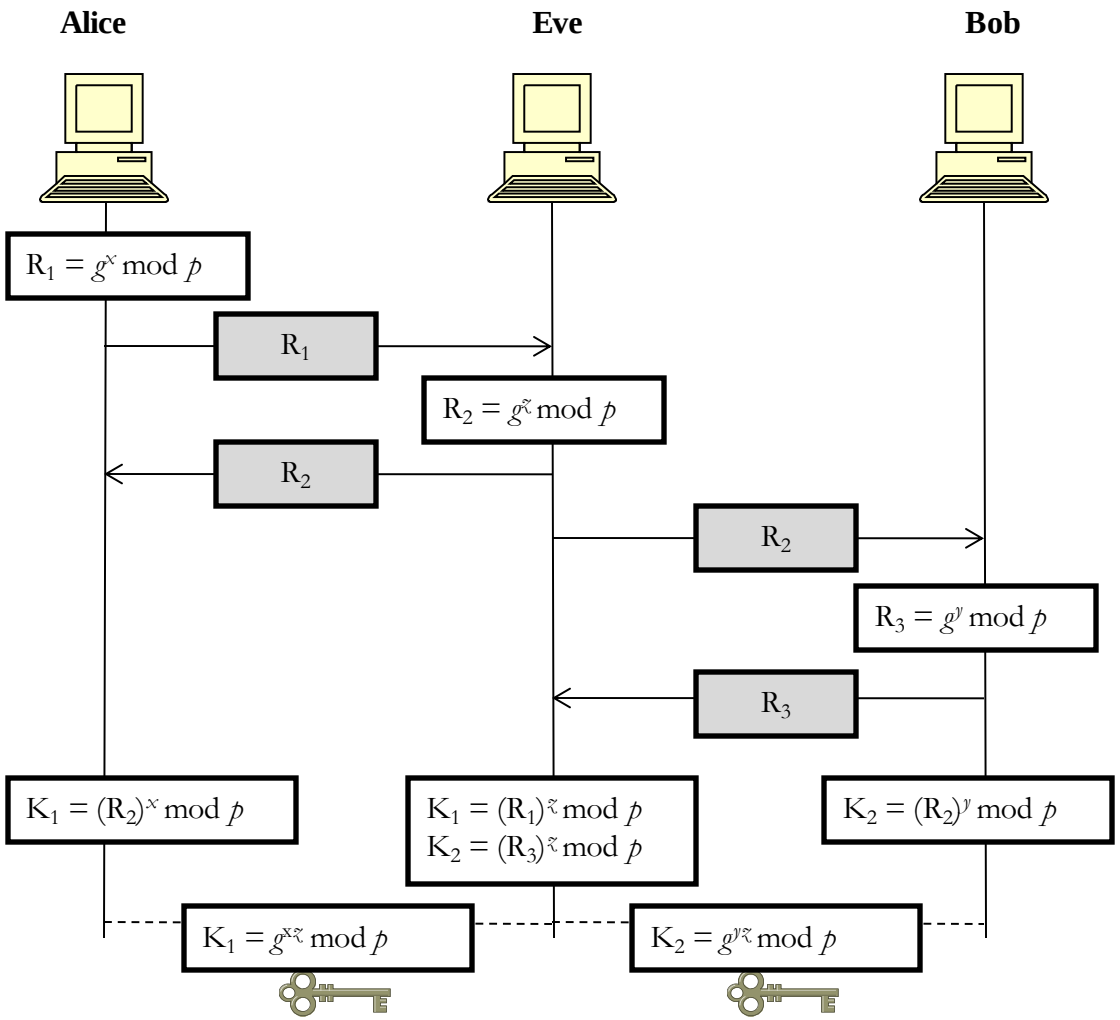
- Man in the Middle Attack
  - Bob, chooses y, calculates $R_3 = g^y \bmod p$ and sends R_3 to Alice and Bob
  - R_3 is intercepted by Eve and never reaches Alice

Security of Diffie-Hellman Key Exchange



- Man in the Middle Attack
  - Alice and Eve calculate $K_1 = g^{xz} \bmod p$, which becomes shared key between Alice and Eve
  - Alice however thinks that it is a key shared between Bob and herself

Security of Diffie-Hellman Key Exchange

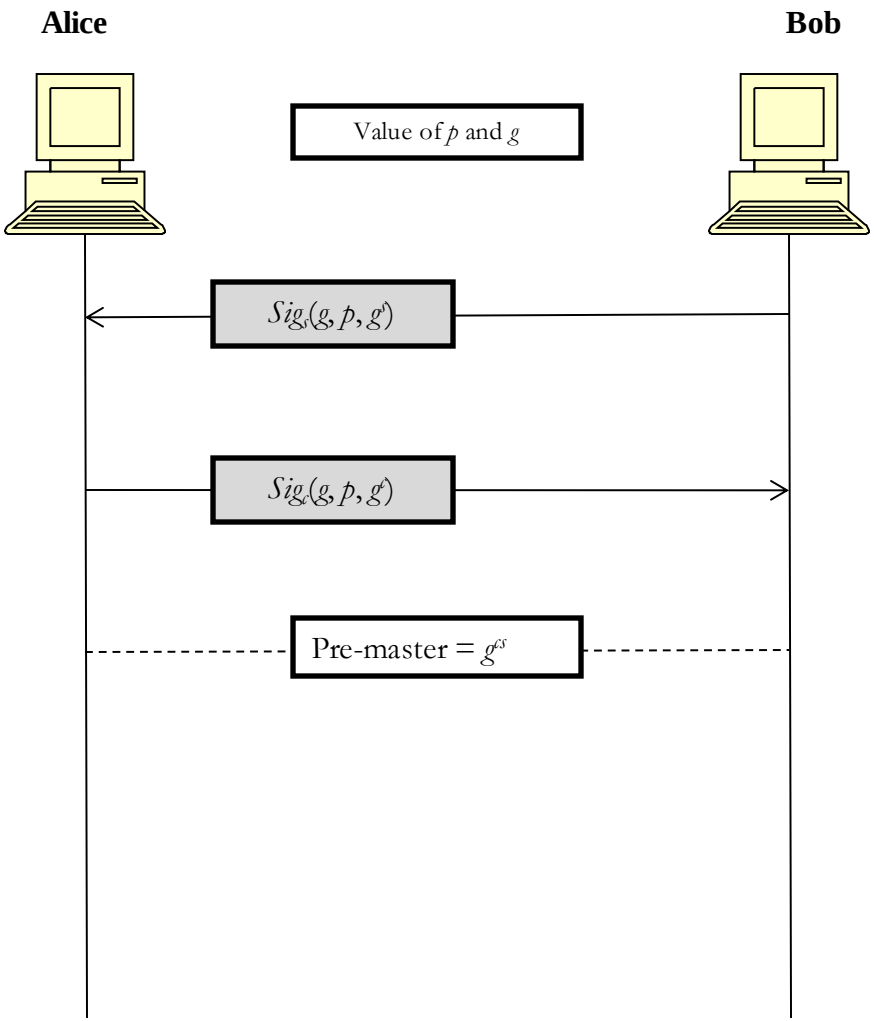


- Man in the Middle Attack
  - Eve and Bob calculate $K_2 = g^{yz} \bmod p$, which becomes shared key between Eve and Bob
  - Bob however thinks that it is a key shared between Alice and himself

Key Exchange Algorithms

- Ephemeral Diffie-Hellman
  - Both parties sends the Diffie-Hellman keys signed by its private keys
- Fixed Diffie-Hellman
  - Diffie-Hellman parameters are signed by certification authority

Ephimeral Diffie-Hellman Key Exchange



- Each party sends a Diffie-Hellman key signed by its private key
- Public keys for verification are exchanged using either RSA or DSS digital signature certificates

Fixed Diffie-Hellman Key Exchange

- All entities in a group prepare fixed Diffie-Hellman parameters (g and p)
- Each entity can create a fixed Diffie-Hellman half-key (g^X)
- Each individual half key is inserted into a certificate verified by a certification authority (CA)
- When the client needs to calculate the pre-master it uses its own half-key and the server half-key received in a certificate

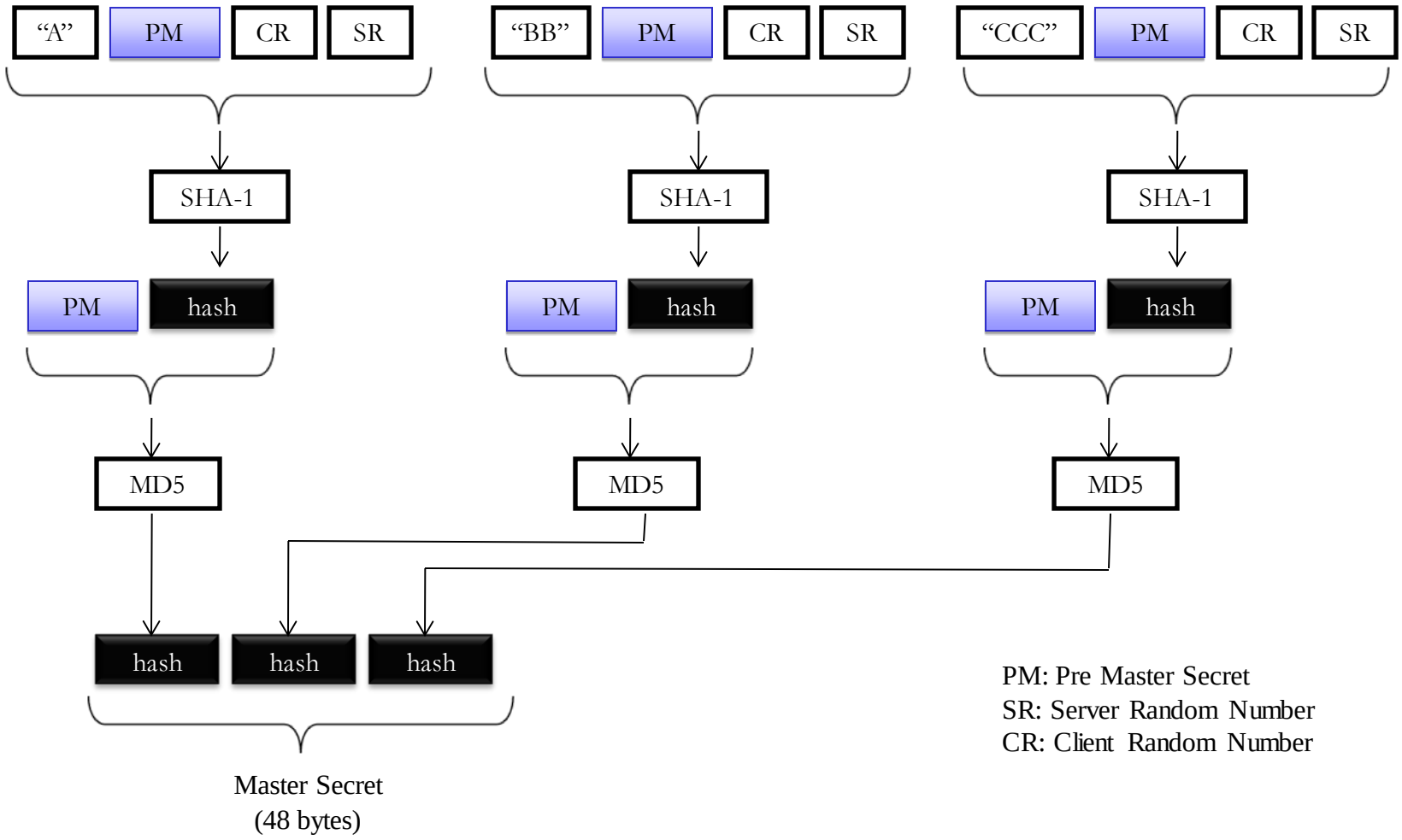
Cipher Suite

- Combination of key exchange, hash, and encryption algorithm defines a cipher suite

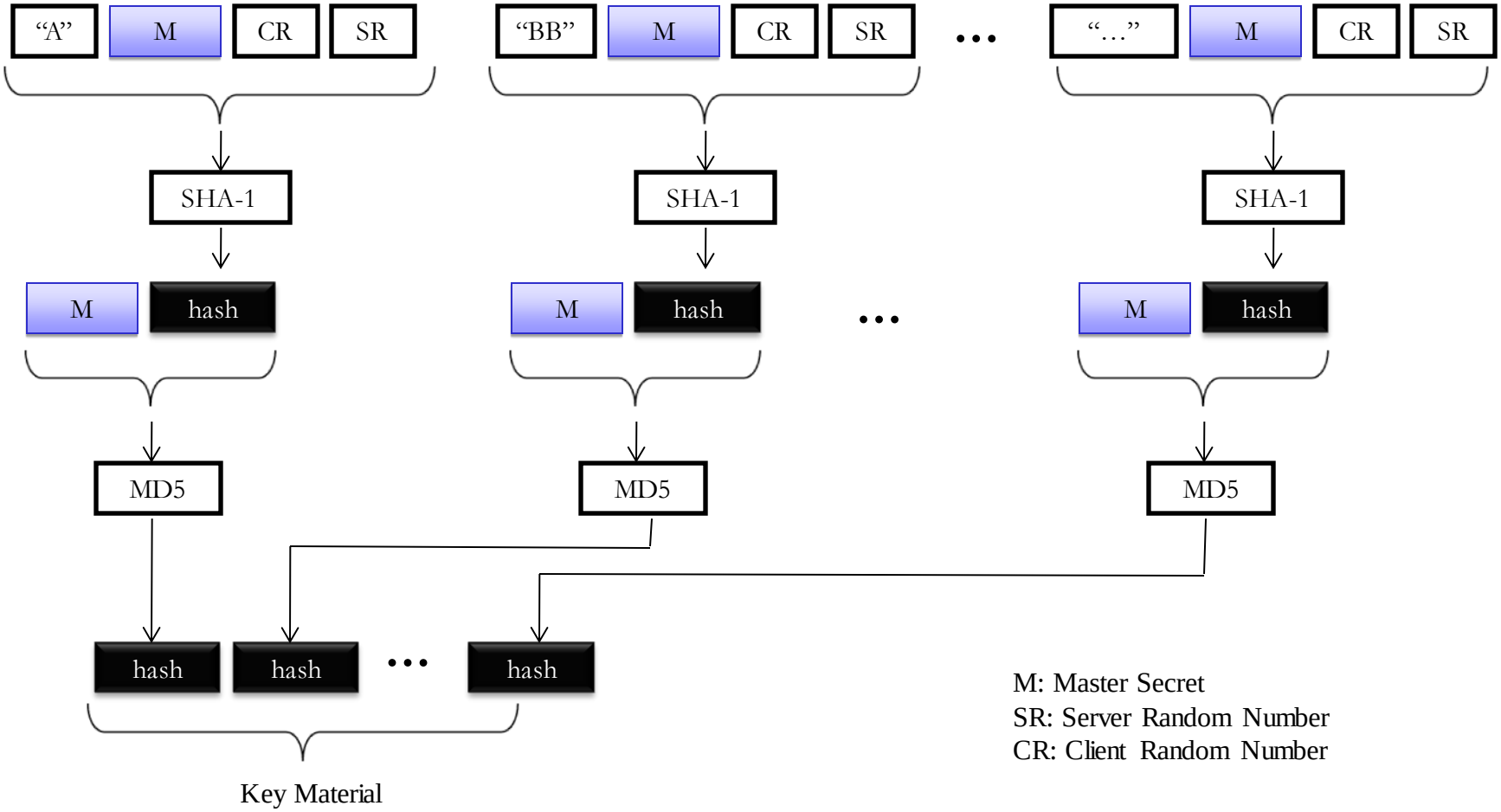
Cryptographic Parameter Generation

- Client and server exchange two random numbers
- Client and server exchange one pre master secret using one of the key exchange algorithms
- A 48 byte master secret is created from the pre-master secret by applying two hash functions (SHA-1 and MD5)
- The master secret is used to create variable length key material by applying the same set of hash functions
- Six different keys are extracted from the key material

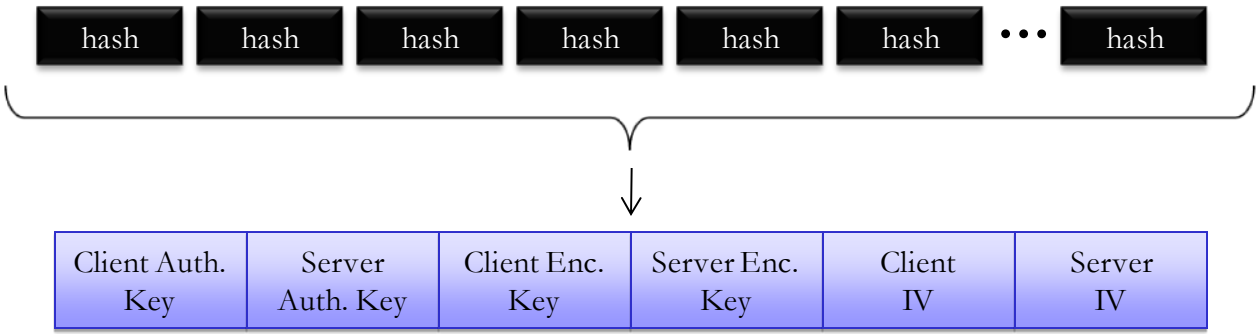
Calculation of Master Secret from Pre-Master Secret



Calculation of Key Material from Master Secret



Extraction of Cryptographic Secrets from Key Material



TLS Handshake Protocol

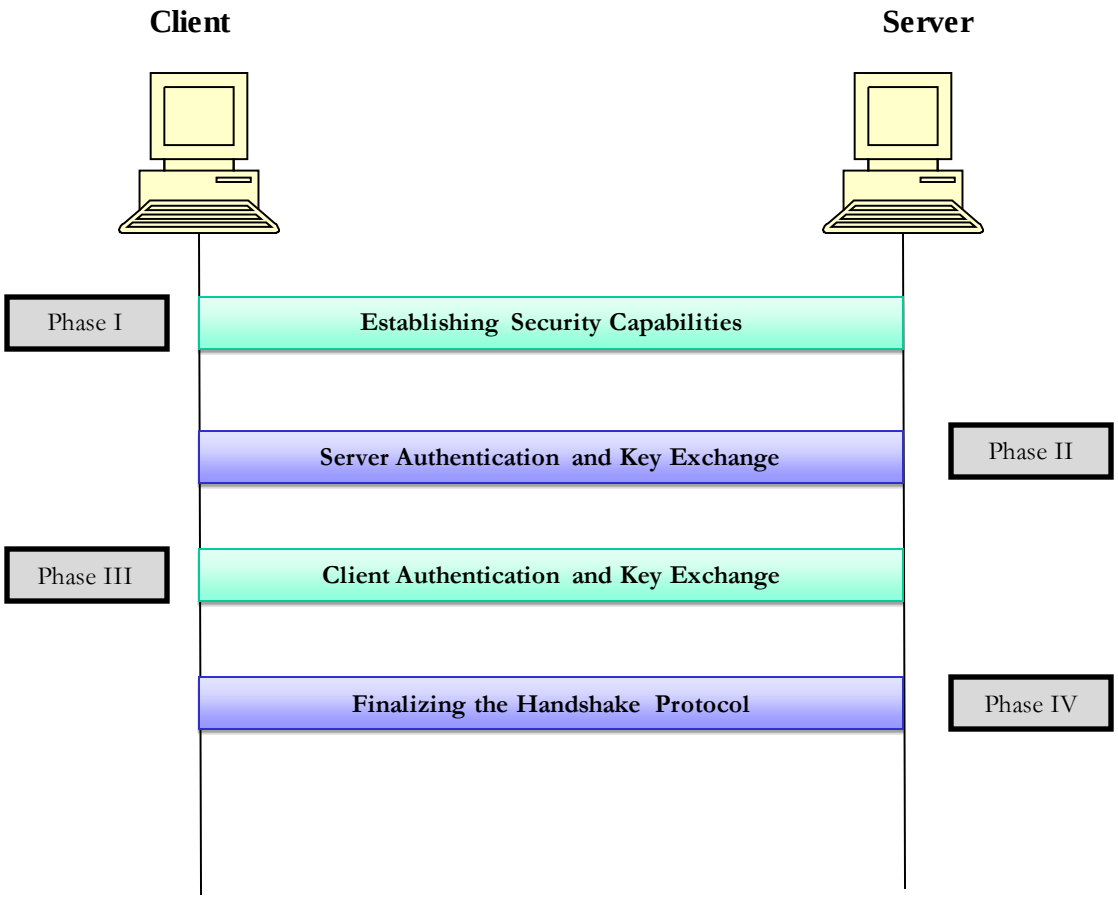
- Allows server and client to:
  - Authenticate each other
  - To negotiate encryption & MAC algorithms
  - To negotiate cryptographic keys to be used
- Comprises a series of messages in phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

Handshake Protocol Message

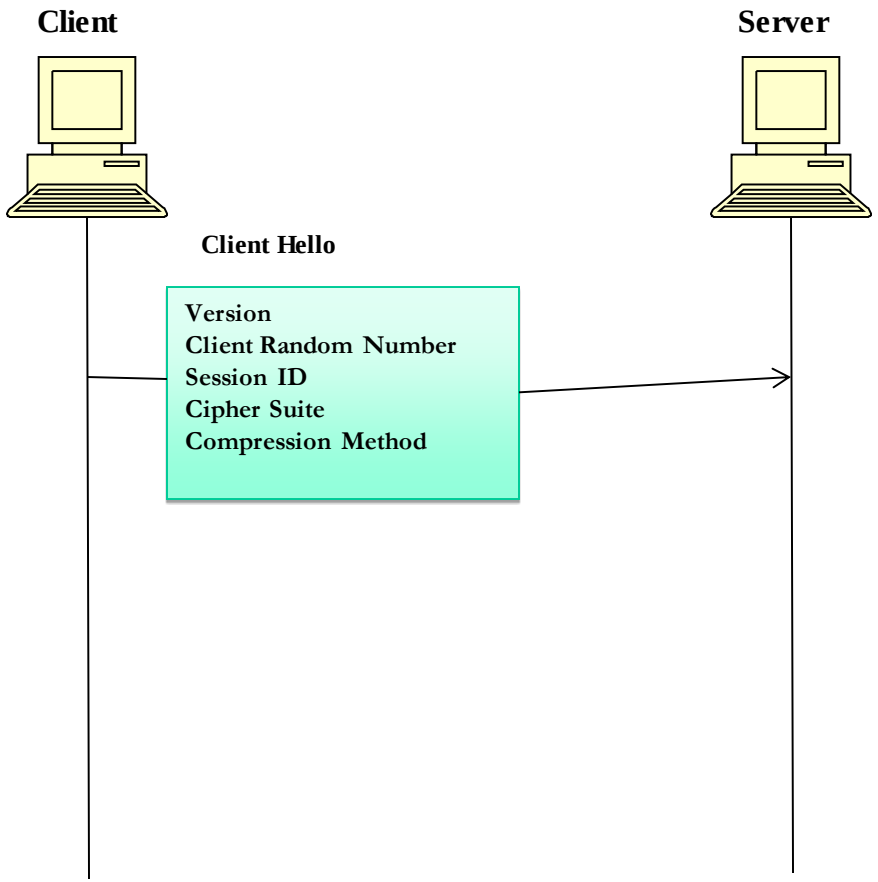
- Handshake protocol message fields
  - Type (1 byte)
    - Indicates one of 10 messages
  - Length (3 bytes)
    - Length of message in bytes
  - Content (>= 0 byte)
    - Parameters associated with this message

Message Type	Parameters
`hello_request`	null
`client_hello`	version, random, session id, cipher suite, compression method
`server_hello`	version, random, session id, cipher suite, compression method
`certificate`	chain of X.509v3 certificates
`server_key_exchange`	parameters, signature
`certificate_request`	type, authorities
`server_done`	null
`certificate_verify`	signature
`client_key_exchange`	parameters, signature
`finished`	hash value

TLS Handshake Protocol

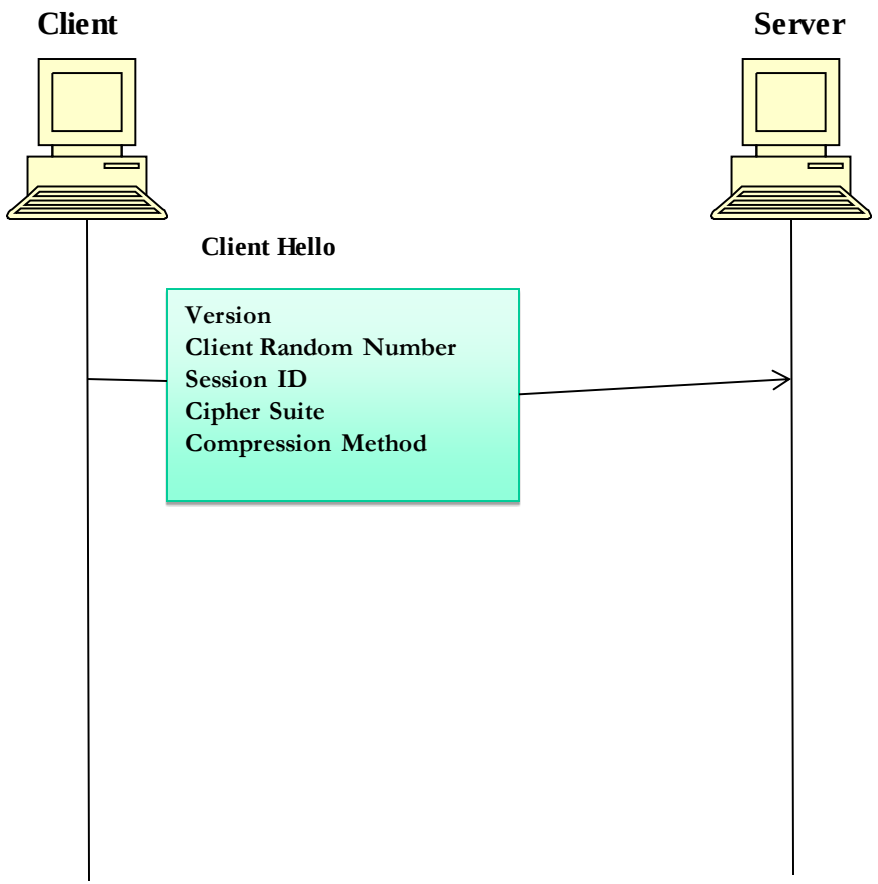


TLS Handshake Protocol
Phase 1: Establish Security Capabilities



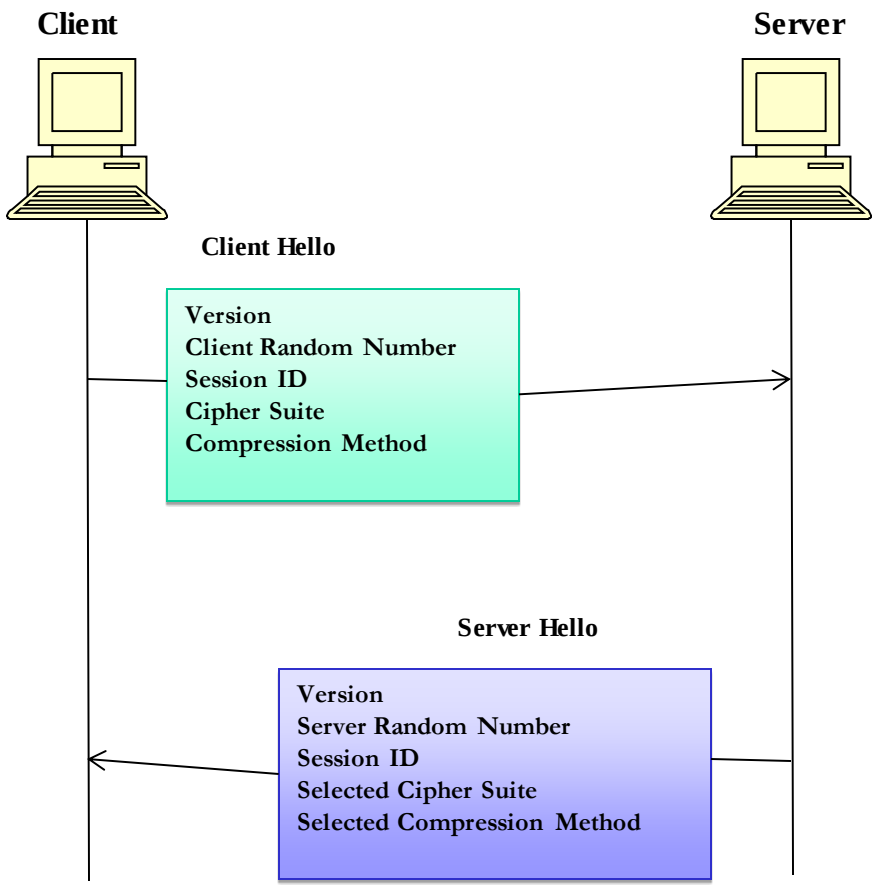
- Used to establish a connection and its associated security capabilities
- Initiated by client which sends a client_hello_message with the following parameters
  - Version
    - Highest TLS version understood by client
  - Random
    - 32 bit timestamp and 28 byte random number
    - Serves as a nonce and used during key exchange to prevent replay attack

TLS Handshake Protocol
Phase 1: Establish Security Capabilities



- Session ID
  - Variable length
  - A zero value indicates that client wishes to establish a new connection on a new session
  - A non-zero value indicates that client wishes to update parameters of an existing session or create a new connection on this session
- Cipher Suite
  - List of elements defining both a key exchange algorithm and a CipherSpec (in decreasing order of preference)
- Compression Method
  - List of compression method that client supports

TLS Handshake Protocol
Phase 1: Establish Security Capabilities

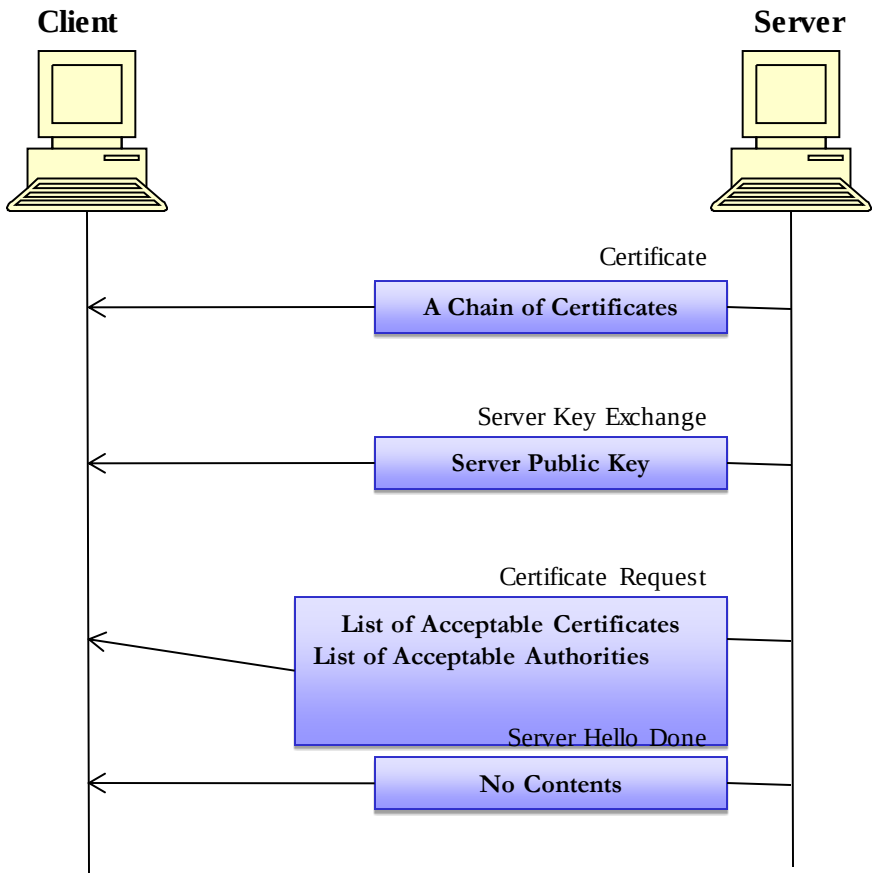


- Client waits for server_hello_message with same parameters
  - Version
    - Lower of the version suggested by client and highest supported by the server
  - Random
    - Generated by server and independent of clients random
  - Session ID
    - Same as that of client's if nonzero
    - A new session id otherwise
  - Cipher Suite
    - Single cipher suite chosen by server from those proposed by client
  - Compression Method
    - Compression method chosen by server from those proposed by client

TLS Handshake Protocol
Phase 1: Establish Security Capabilities

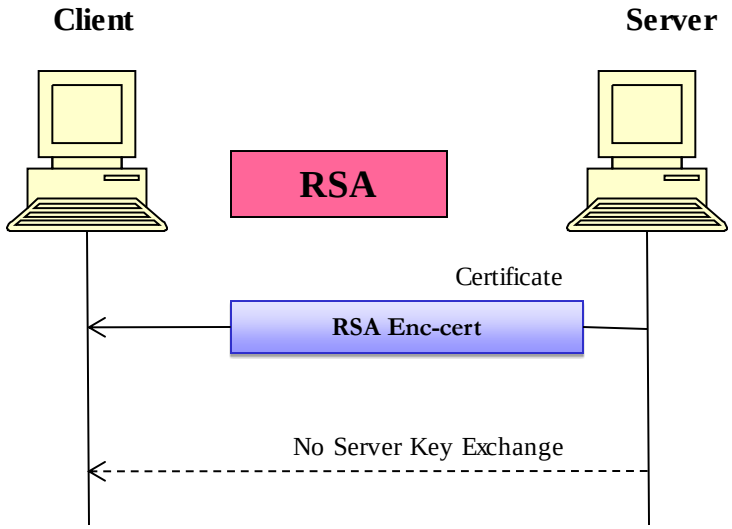
- First element of the CipherSuite parameter is the key exchange method
  - RSA
  - Fixed Diffie-Hellman
  - Ephemeral Diffie-Hellman
  - Anonymous Diffie-Hellman
  - Fortezza
- Second element of the CipherSuite parameter is the CipherSpec which incudes
  - CipherAlgorithm
  - MACAlgorithm
  - CipherType

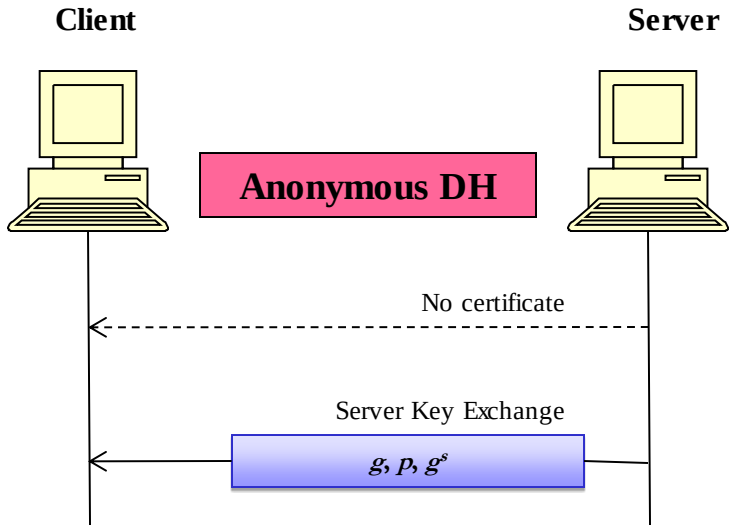
TLS Handshake Protocol
Phase 2: Server Authentication and Key Exchange

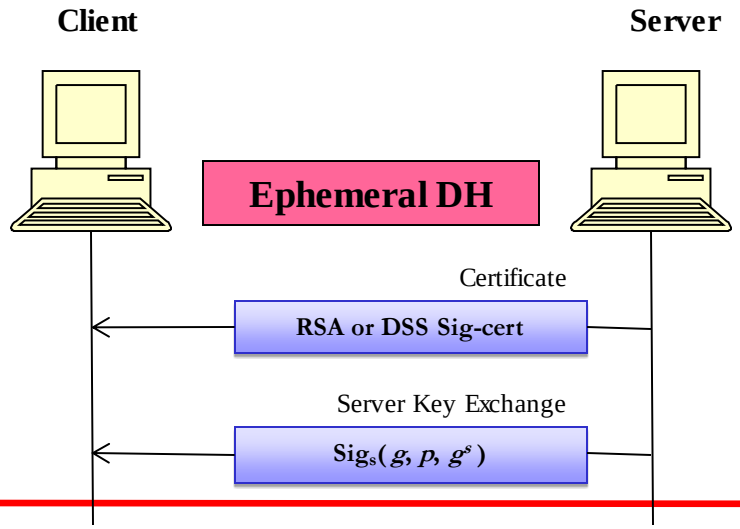


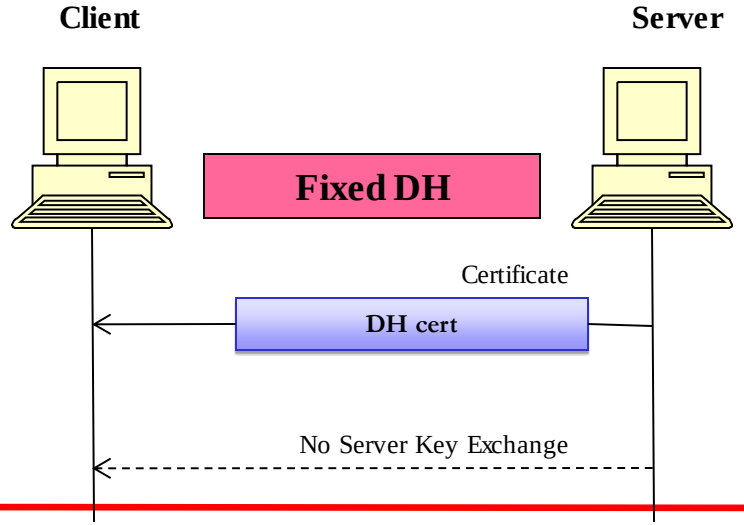
- Server sends its certificate
  - Required if server needs to authenticate itself
  - Not required if the key exchange algorithm is anonymous Diffie-Hellman
- Server sends server_key_exchange message
  - Required if the key exchange method selected is
    - RSA (Server has signature only RSA key)
    - Ephemeral Diffie-Hellman
    - Anonymous Diffie-Hellman
  - Not required if the key exchange algorithm is fixed Diffie-Hellman
- Server sends certificate_request message
  - Required if client authentication is needed
- Server sends server_done message

TLS Handshake Protocol
Phase 2: Server Authentication and Key Exchange

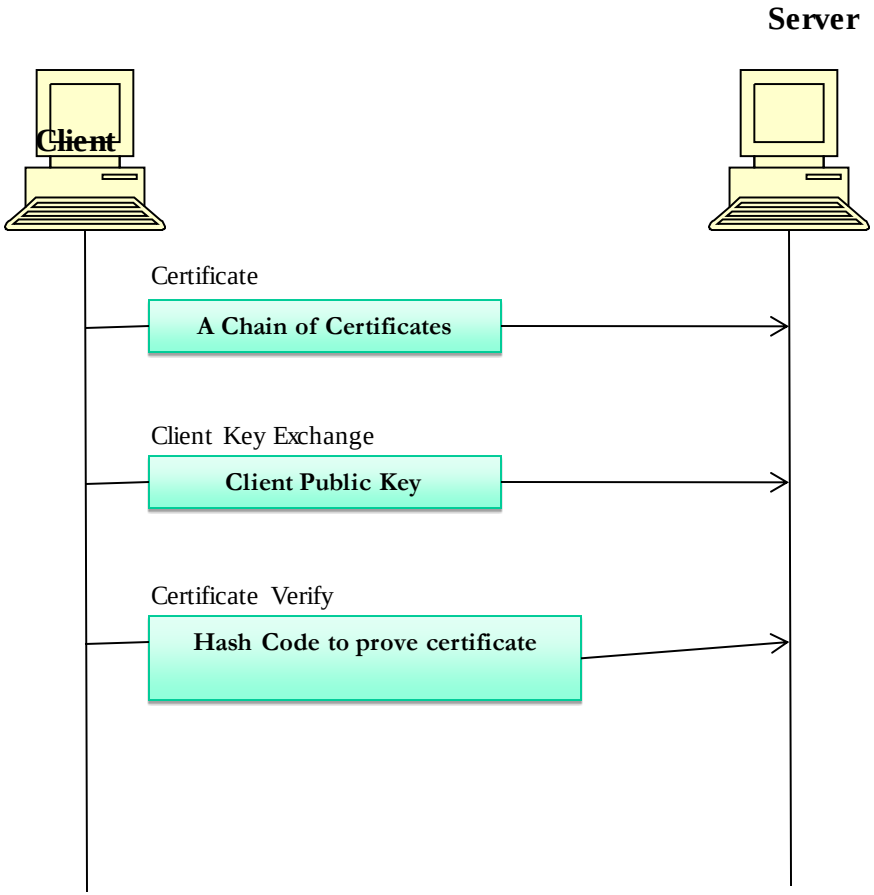






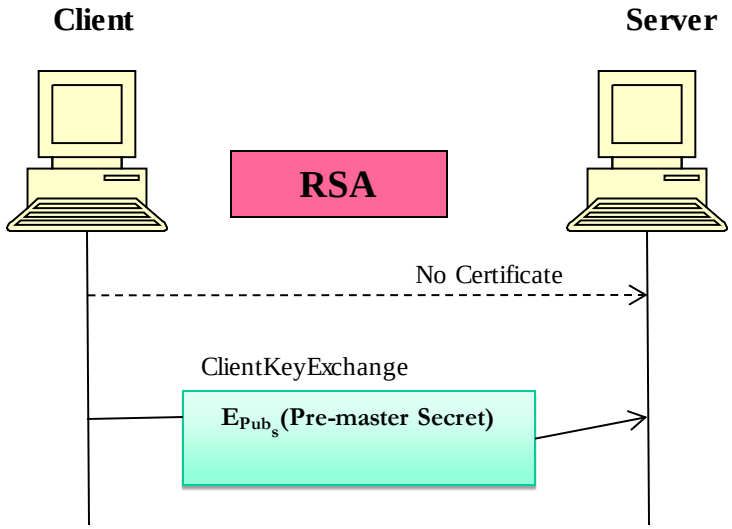


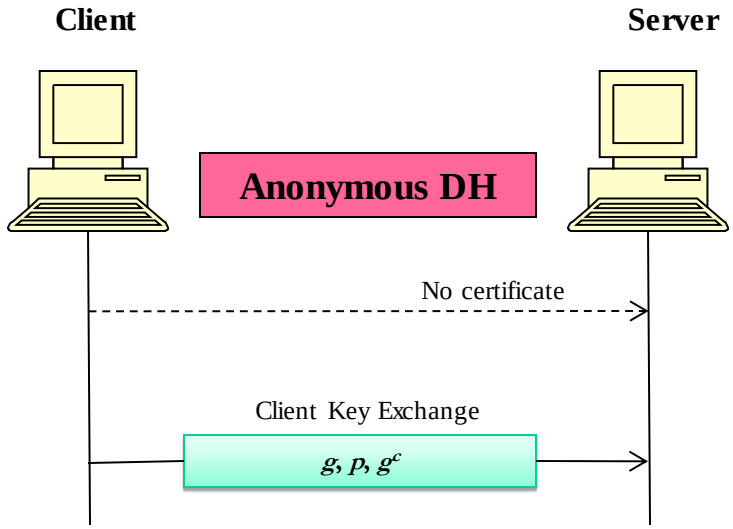
TLS Handshake Protocol
Phase 3: Client Authentication and Key Exchange

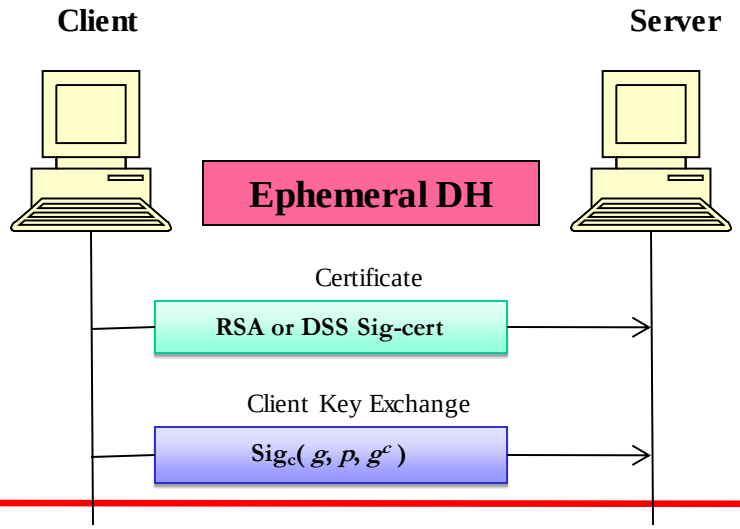


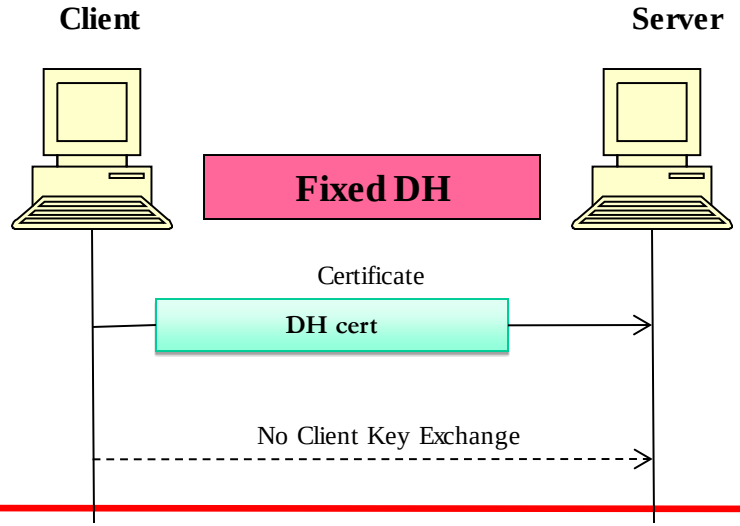
- Client sends its certificate (if requested by server)
  - If no suitable certificate is available, the client sends a no_certificate alert
- Client sends client_key_exchange message
  - Depends on the type of key exchange method selected
- Client sends certificate_verify message to provide explicit verification of a client certificate
  - Only sent following any client certificate that has signing capability
  - Signs a hash code based on preeding messages

TLS Handshake Protocol
Phase 3: Client Authentication and Key Exchange

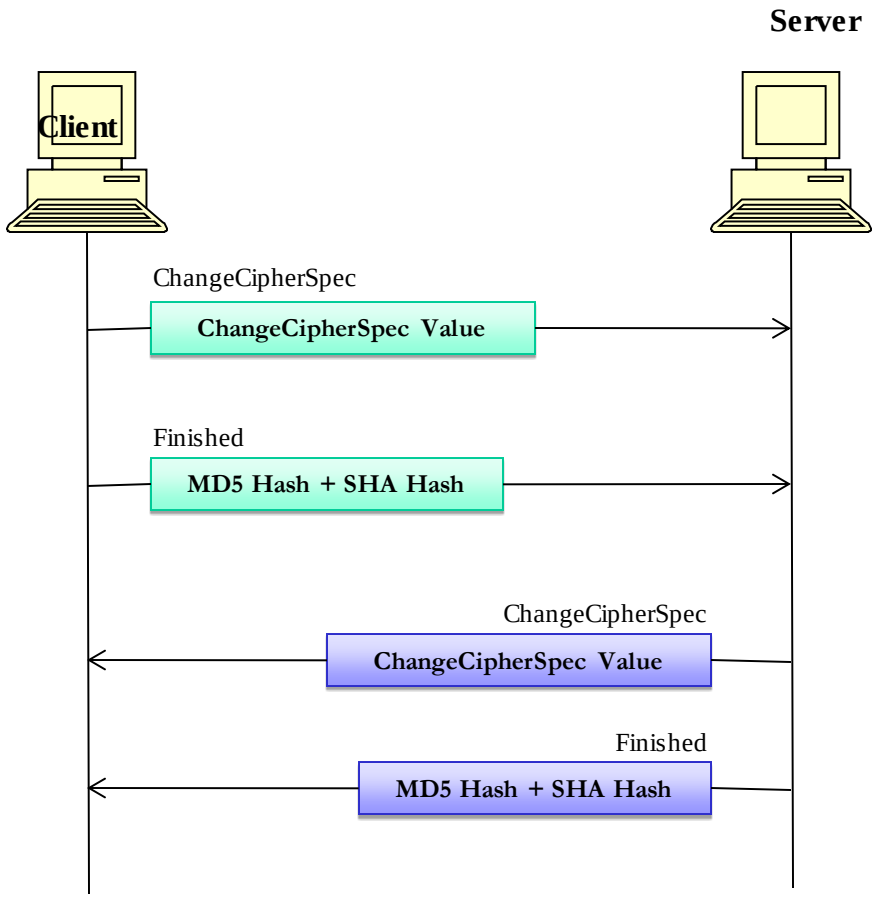






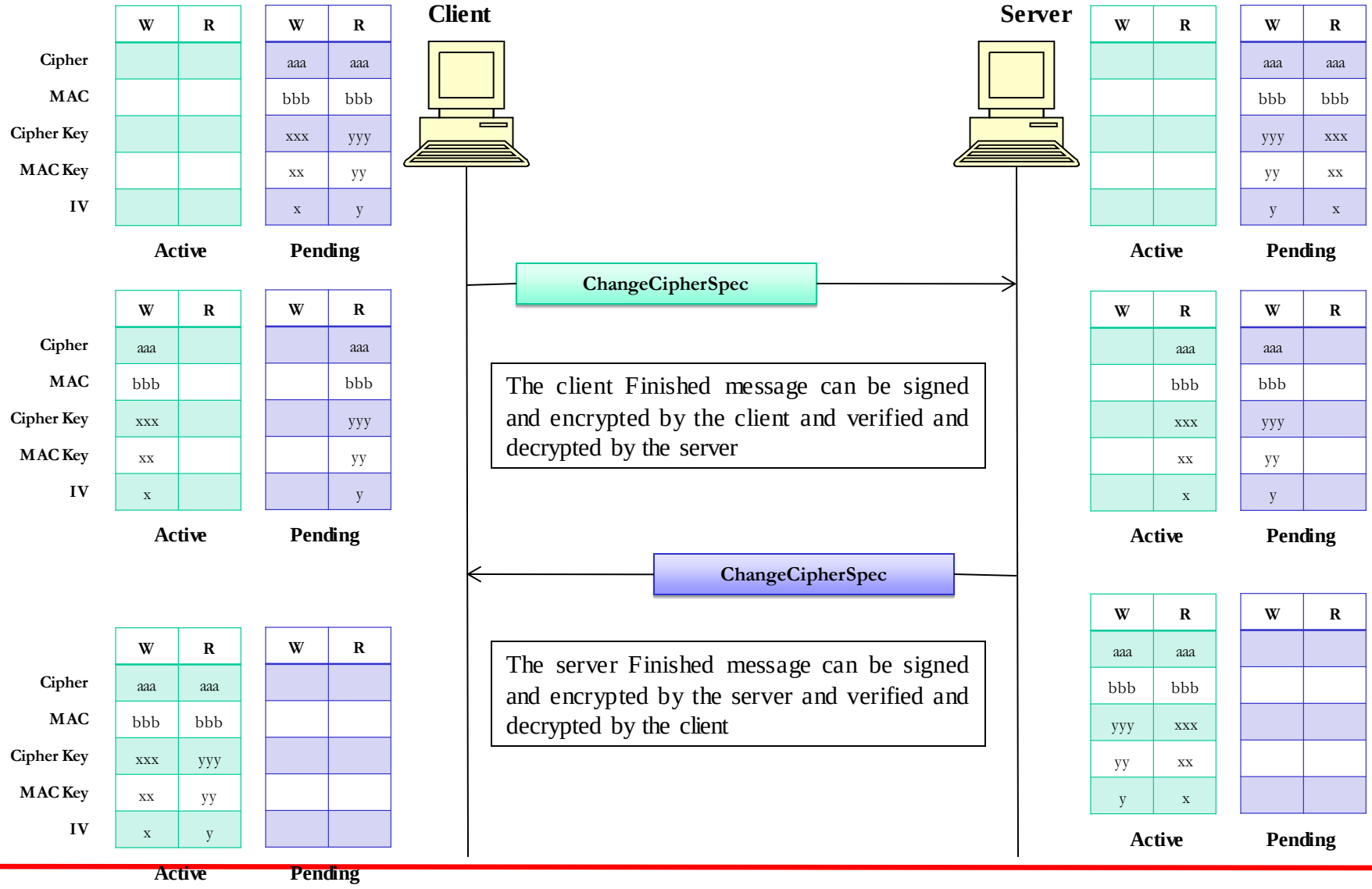


TLS Handshake Protocol
Phase 4: Finish

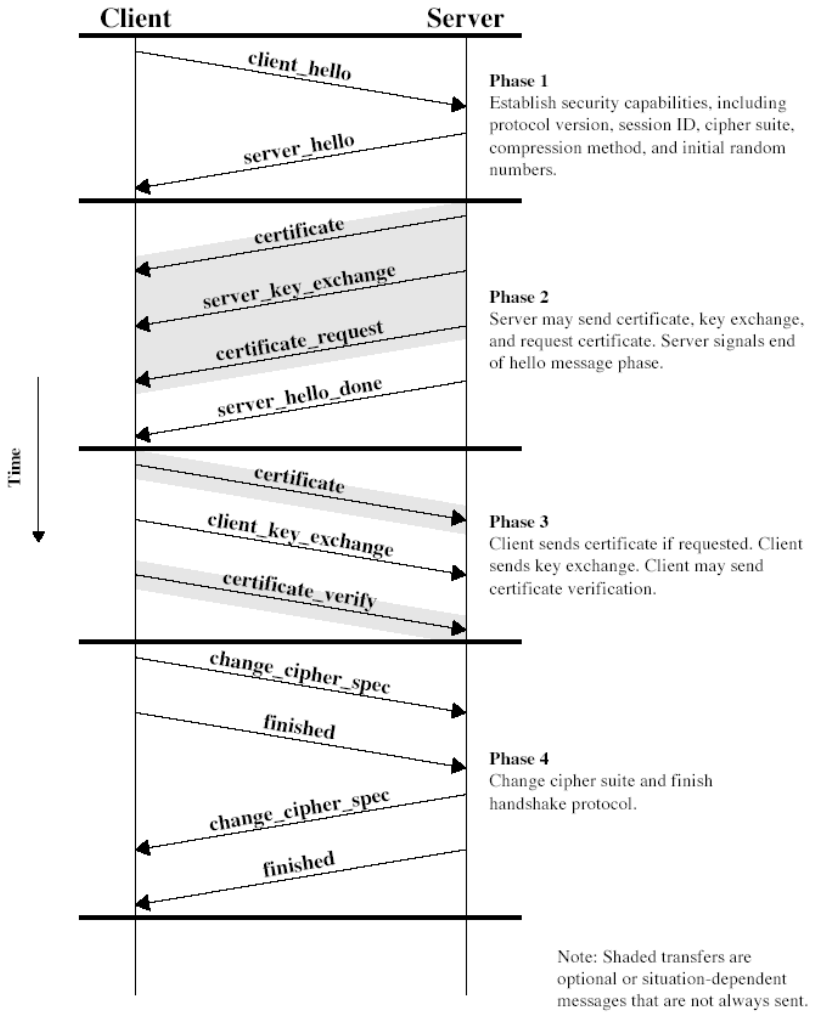


- Client sends change_cipher_spec message
  - Copies pending CipherSpec into current CipherSpec
- Client sends finished message under the new algorithms, keys and secrets
- Server sends change_cipher_spec message
  - Copies pending CipherSpec into current CipherSpec
- Server sends finished message under the new algorithms, keys and secrets

TLS Handshake Protocol
Phase 4: Finish



TLS Handshake Protocol



TLS Record Protocol

- Provides two services for each TLS connection
  - Confidentiality
    - Using symmetric encryption with a shared secret key defined by Handshake Protocol
    - Block Ciphers
      - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza
    - Stream Ciphers
      - RC4-40, RC4-128
    - Message is compressed before encryption
  - Message integrity
    - Using a MAC with shared secret key
    - Similar to HMAC but with different padding

TLS Record Protocol: Operation