17. Chapter

* IP sec

→ tends to assume that secure requirements are a function of a particular system

→ all applications within the system need the same security service

* Network security

1. security protocol
2. application level security. (kerberos)
3. core protocol.
4. parallel security protocol. [ex : kerberos] (no encryption/decryption)

* SSL Limitations

- Fundamental protocol limitations (no support for UDP, non-repudiation) — DTLS. recently. works.
- inherits some weaknesses from tools. (some tools are attacked successfully)
- ssl itself have their own shortcomings and limitation (after connection no guarantee the security)

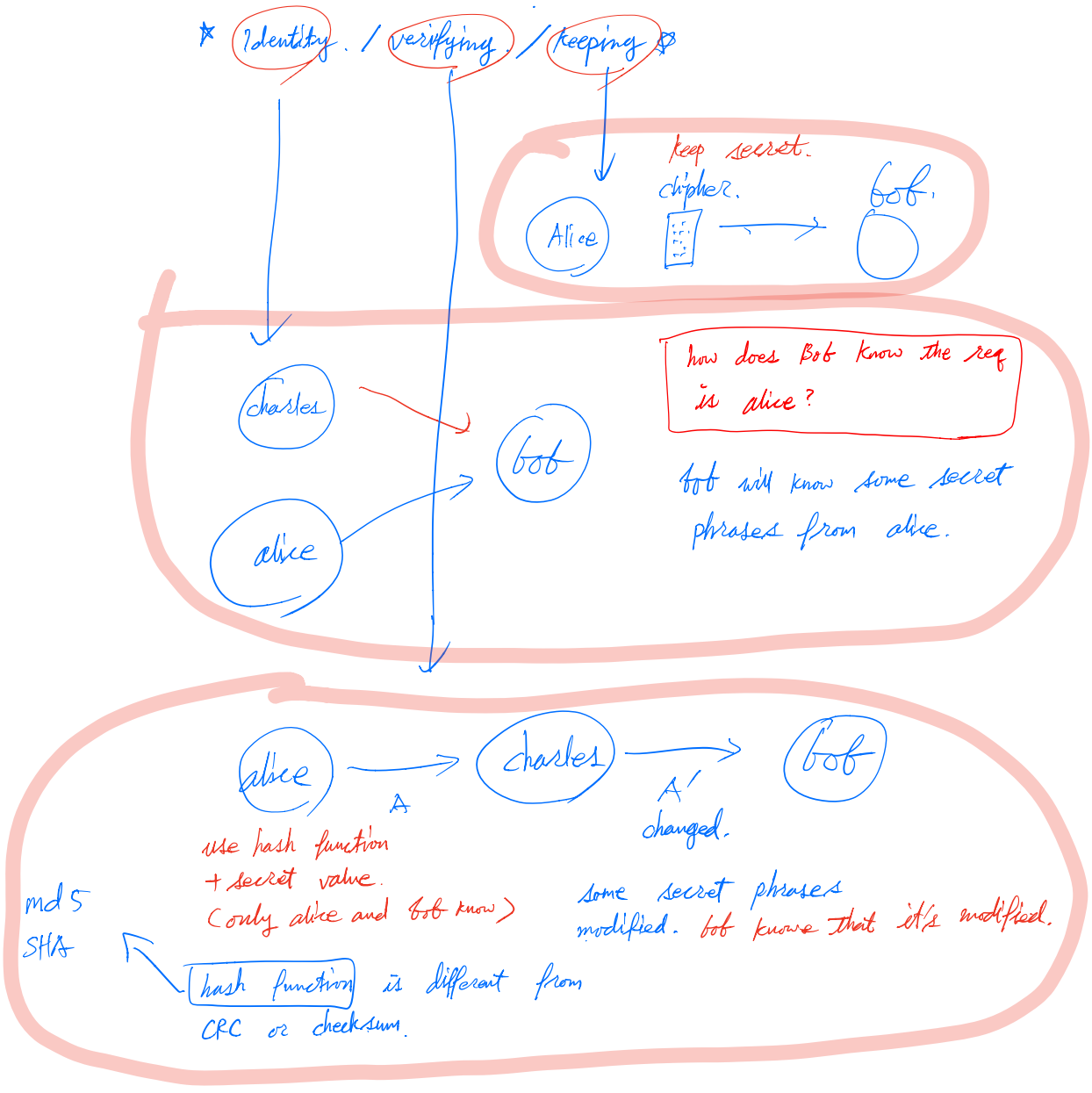
2 chapter

- web → needs security?

* Why Security needs?
  * web commerce
* Cryptography usage

* Cryptography means ...
  * "secret writing in greek.

* Identifying / verifying / keeping *



Type of Cryptography

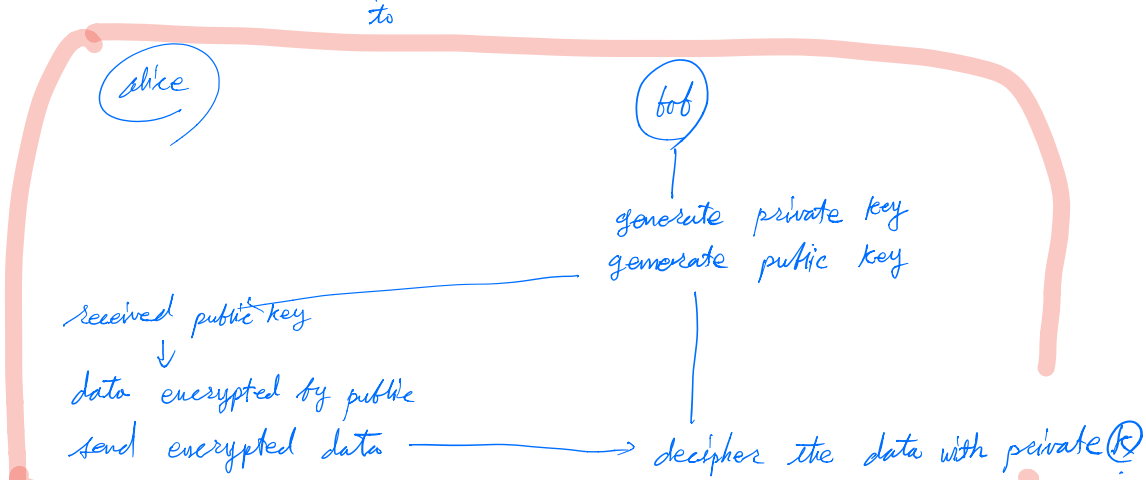
Encryption algorithm = chipher.

- Secret key Cryptography → symmetric encryption
  - key size is the great part of security quality.
  - chiper — stream cipher : a byte a time / no limit.
    - block cipher : fixed size (less vulnerable)
      - by far more common.
      - need padding to know it's end.
      - need initialize vector of dummy to begin enc. process.
        - → prime number.

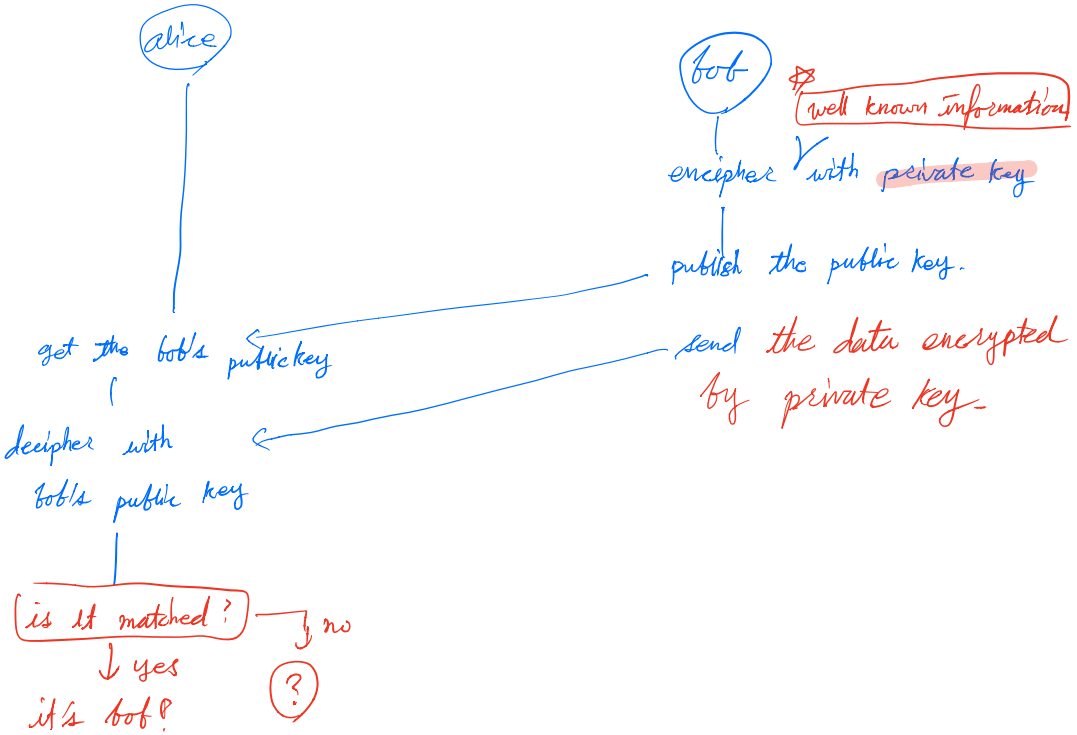
symmetric algorithm

DES	block
3DES	block
RC2	block
RC4	stream

- public key cryptography.
  - use asymmetric encryption (two parties use separated keys.)
  - basis.: much easier to generate than to solve



- public/private key works in reverce
  - providing identity (DSA)



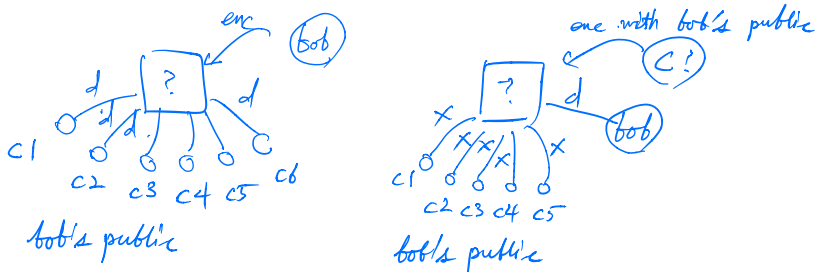
Bob has proven his identity to alice (digital sign)

(only bob can encrypt the data)

? what if someone who has bob's publickey encrypts the data with bob's public key? (pretends bob)

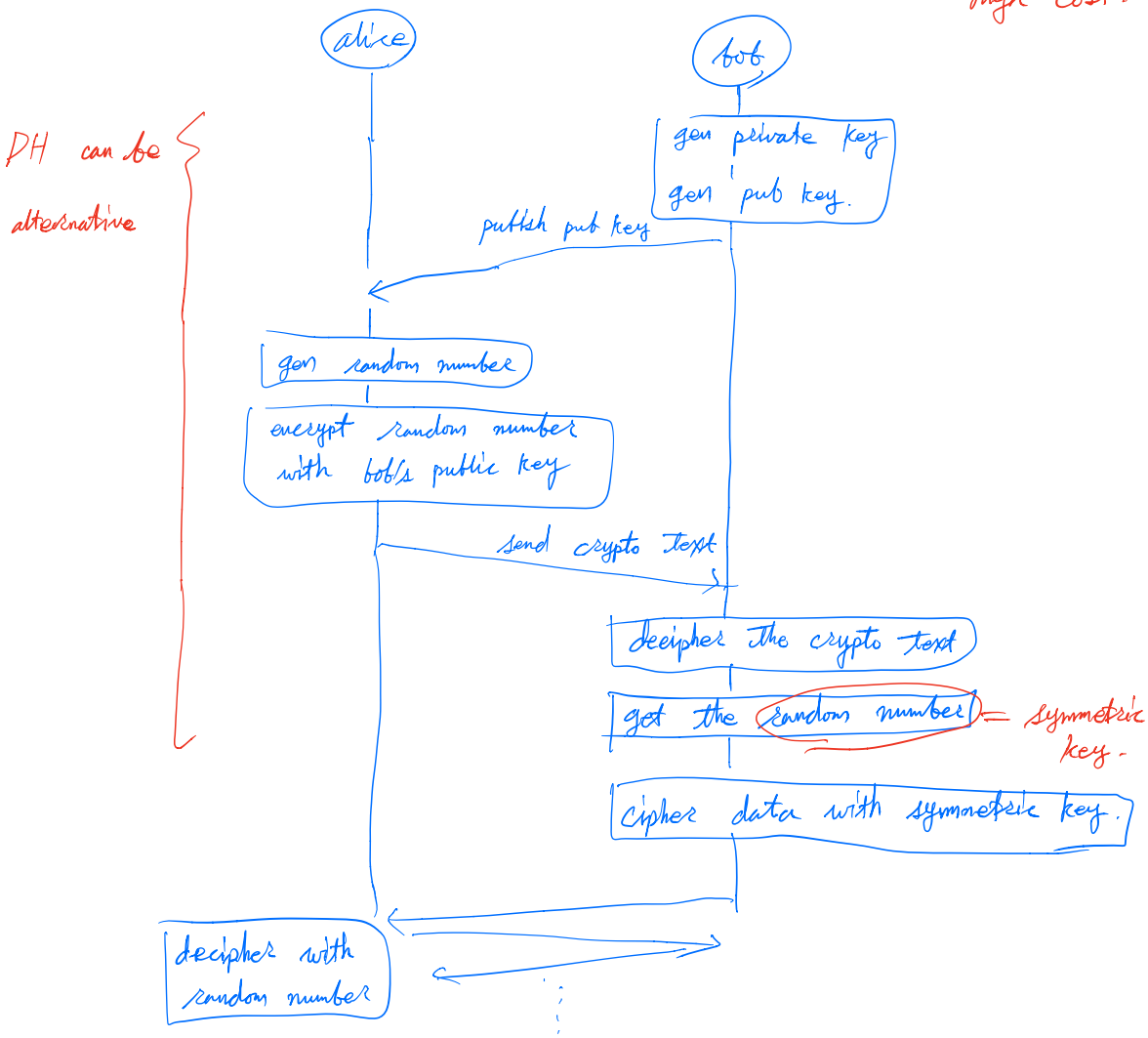
∴ no one except bob can not decrypt the data

It's not useful for providing identity of bob itself.



Combine Secret and public Crypto

- encryption operation is Complex (strain on system)
  - ↳ solution : first time use public key algo to share symmetric key algo.
    - ↳ high cost.

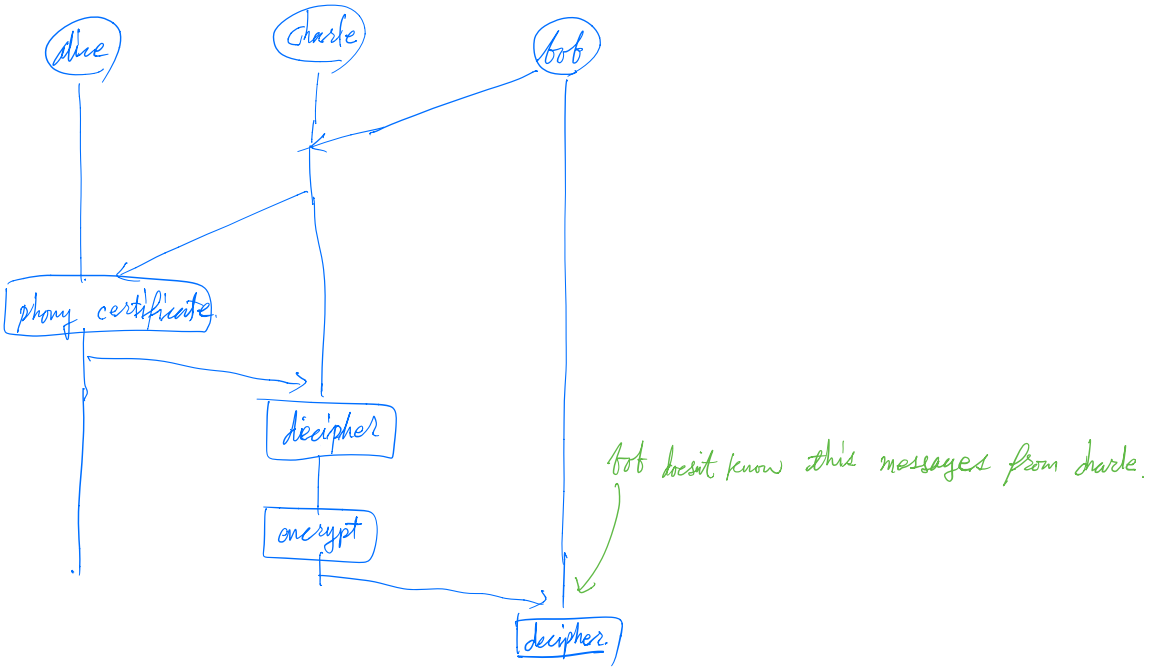


DH Diffie-Hellman (asymmetric key algorithm)

* it's public key algorithm but It cannot be used for encryption or digital signature
* so, It allows to exchange key by using only public message

key management.

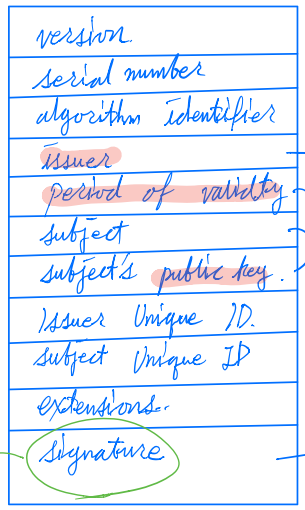
→ if we use public key exchange, someone steals (could) or interveres.



It's critical to the Secure Socket Layer and Web commerce.

Public key Certificate.

* 3 things important characteristics
  1. subject name.
  2. assert key information about the subject
  3. are issued by a trusted organisation.
* Public key's contents.



→ identifies the organization that has issued cert.

→ expire time.

→ issuer's signature.

* issuer creates this signature by encrypting a hash of the certificate with its private key (= issuer)

this ?

Cipher ([data] hash(cert) , [key] issuer's private)

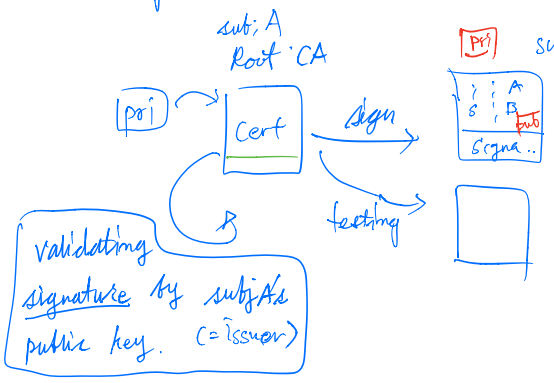
more detail.: the issuer creates the signature using its own private key
(the certificate itself contains the subject's public key.

so, it can be verified by issuer's public.

Certificate authorities

* Issuer's public key certificate is traditionally known as CA.
* This makes a community of users for establishing
* certificate authorities are themselves frequently identified by their certificates

if subject == issuer : Root CA.



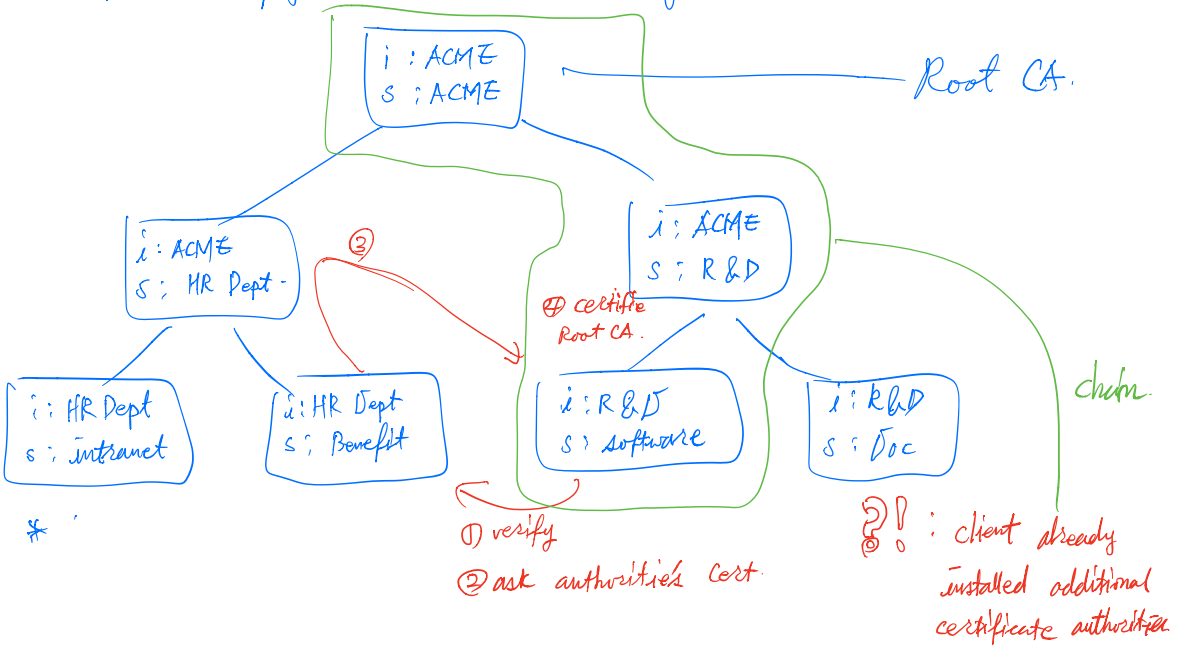
? if you need to validate B's cert, you need A's certificate

sign : Encrypt certificate with ISSUER's Private key

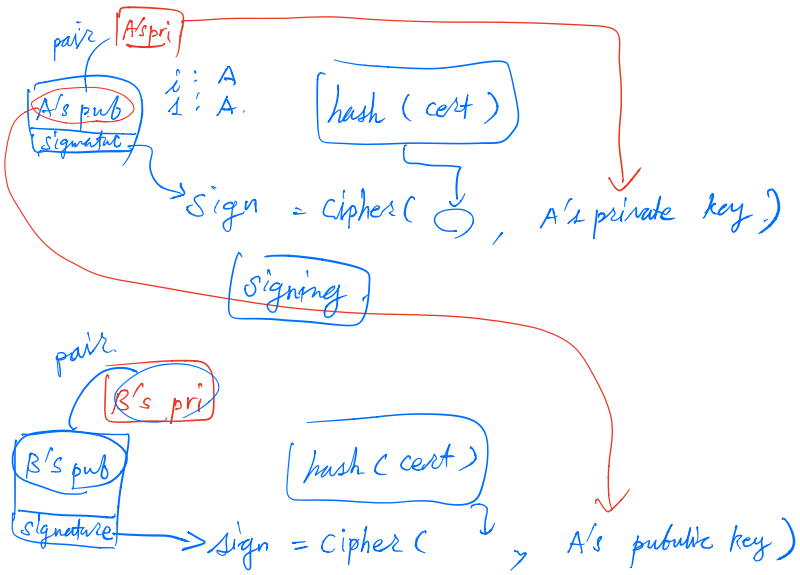
Testing : Decrypt Certificate with ISSUER's Public key.

if the certificate is verified, issuer is trustworthy. and subject's public key is safe.

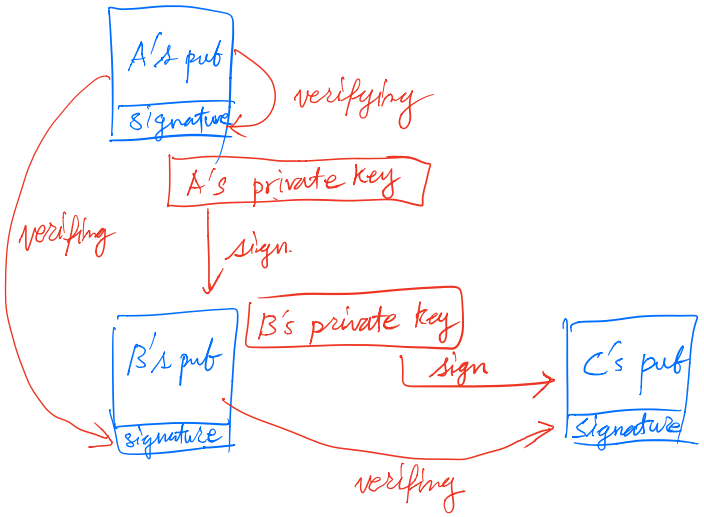
Certificate simply three level hierarchy



The way of generating chain.



verifying



the process of C's verifying certificate.

① find Root CA.

② A → B verifying

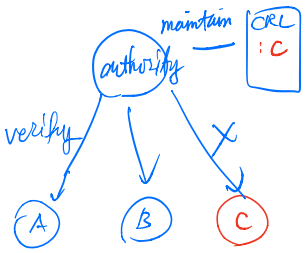
③ B → C verifying.

Certificate Revocation List

There are many cases about certificates.

1. Root CA issues wrong certificate.
2. subject's private key reveals..
3. public key no longer safe

CRL : is a list of certificates that authority has previously issued but no longer consider valid



3 chapter

Purpose

* ssl establishing an encrypted communication channel.
* ssl authenticating / resuming previous session
* ssl protocol message. format. and rules.

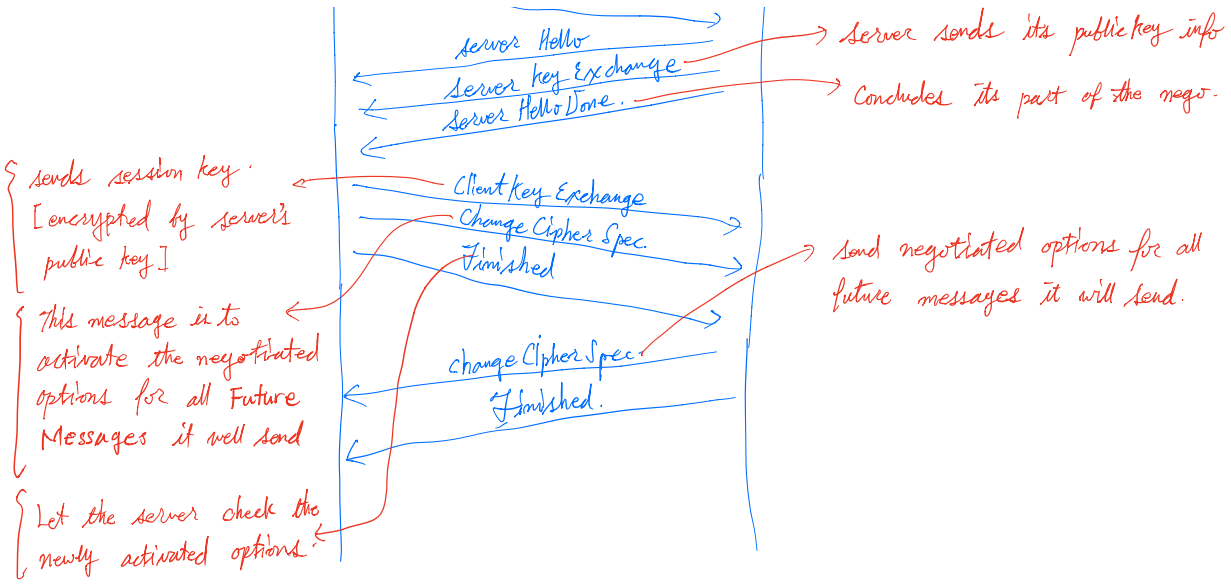
ssl Role

- Client – server model. [each role is very different]
  - Client : initiator. [the proposing a set of ssl option]
  - server : responder [select from client's the proposed options]

ssl messages

- alert
- ChangeCipherSpec
- finished
- HelloRequest
- ApplicationData → actual information
- Client Hello
- Client Key Exchange
- Server Hello
- Server Hello Done
- Server Key Exchange
- Certificate → a message that carries the sender's pub key.
- Certificate Request
- Certificate Verify

Establishing Encrypted Communications.



Client Hello :

- version : highest version of SSL that client supports [SSL : 3.0, TLS : 3.1]
  [server assumes that all version of SSL up to this support]
- Random Number : [28 bytes] to seed for critical cryptographic calculation
  [4 bytes] to ensure that client never sends the same value.
  [protect from an imposter copying old SSL messages

use pseudorandom number for cryptography.

- session ID. : would be empty or. something used before.
- Cipher Suites : exact algorithm and key size.
- Compression Methods: list all of the various data
  Client may use compress the data
  if two parties are going to employ the data compression for a communication,
  It's important that they compress data before Encrypting. (some security issues)

Server Hello

- version : determines the SSL version will use. [client ≥ server]
  [after client may abandon the communication].
- Random Number : same meaning of client.
- Session ID : server may contain a value unlike the Client Hello.
  This value in this case uniquely identifying
  this particular SSL communications.
  [used for speeding up the ssl negotiation process].
  ☆ server can decide to allow reuse SSL. ☆
- CipherSuite (singular) : The server must select a single cipher suite from among those listed by the client
- Compression Method (singular) : Choose one from among Client suggests.

Server Key Exchange : send public key itself

☆ the Server key Exchange message is transmitted without encryption so, only public key is safe.

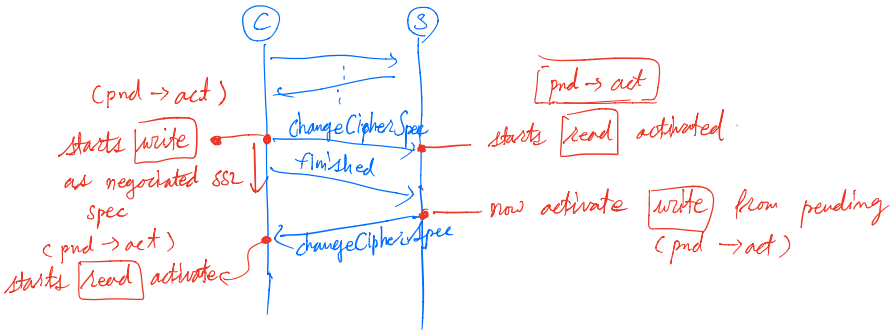
Server Hello Done : Let Client Do next step.

Client key Exchange : send to server key information (symmetric encrypt algo)
The message is encrypted with server's public key.
only real server can decrypt it.

Change Cipher Spec : to explicitly indicate that the security service was be invoked

☆ SSL defines "write" and "read" state.

☆ after sending this message, Client's write state ACTIVATED
(all message will be writed by Encrypt, MAC, Key) ...
↓
(Message Authentication code)



Finished : allow both systems to verify that negotiation has been successful.

☆ Finished message is subject to the negotiated cipher suite.

☆ This message contains a crypto graphic hash of important information

CloseAlert : helps systems detect attack / it's ending session.

☆ SSL authenticate server. Certificate

- Two messages are different. =. 'certificate', 'Client Key Exchange'
- % authenticate its identity sends 'certificate' message instead of "server key Exchange."

☆ Client has the responsible to make sure its trusty

- certificate signature,
- validity time
- revocation status

☆. Client browser knows "well known CA." in advance.

☆ the respected certificate include Internet Domain name of webserver. in cert.

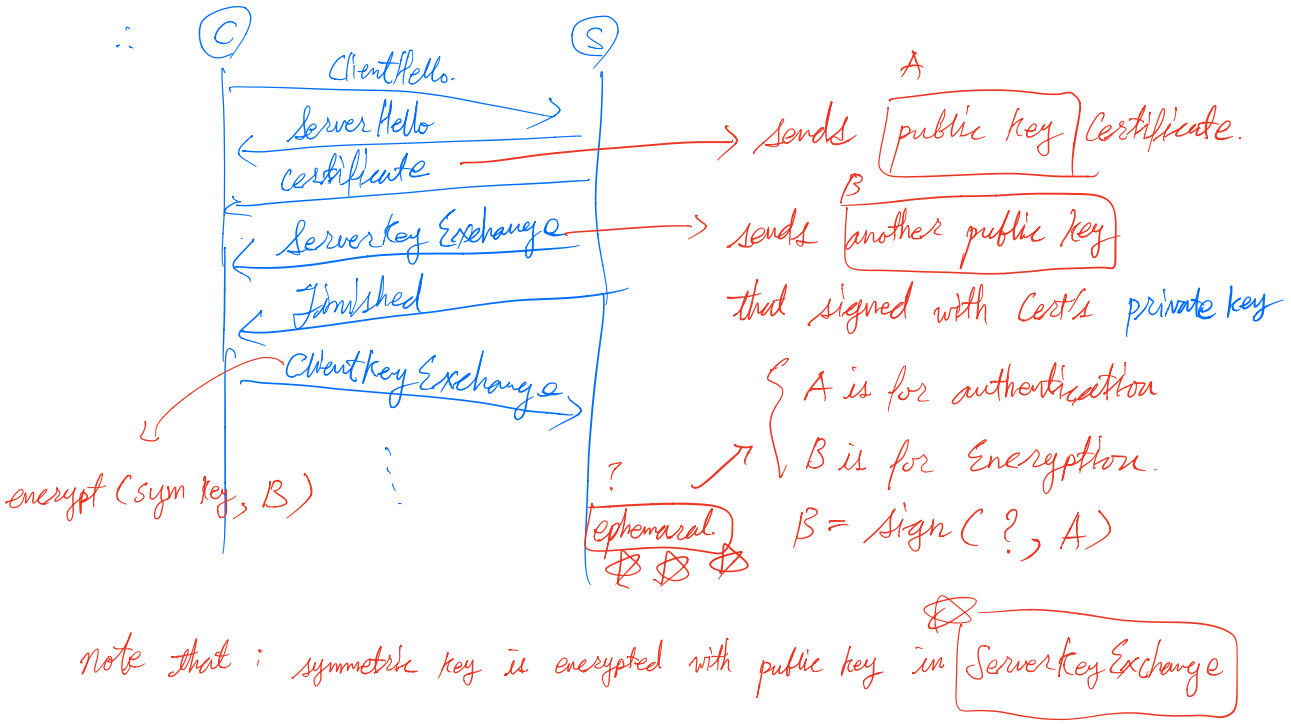
C web browser will check the domain.

Client Key Exchange.

- instead of 'ServerKeyExchange' client uses "certificate" 's public key.
- It means server must contain ☆ private key.

Separating Encryption from Authentication

- Some reason of legal one.
  - System prefer to use longest key length (US not impose the same restriction on key for DSA)
  - So, encryption key doesn't use this.



note that : symmetric key is encrypted with public key in ServerKeyExchange ☆

Certificate Request.

- Server could require this to authenticate client.
- Client simply complies with server's wishes

☆ this message also follow any ServerKeyExchange message

Components.

- CertificateTypes : certificate types acceptable to Server.
- DistinguishedName : distinguished Names of certificate authorities acceptable to server

Certificate (Client → Server)

- if this meets server's criteria, it responds with NoCertificateAlert
- Server can chooce ignore or alert.

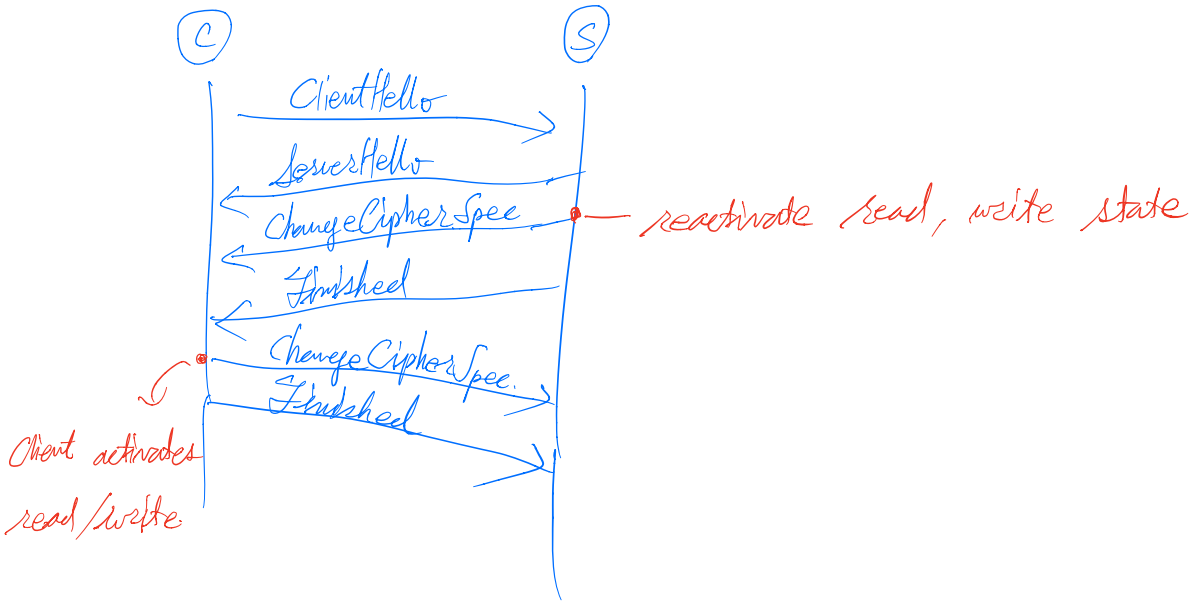
CertificateVerify.

- Client should prove that it poseses the private key corresponding to the certificate's public key.
- For its proof. Send this message.

- digitally signed Cryptography hash of information both available. = Server can verify it
- this message follows ClientKeyExchange because this message needs the cryptographic value.

Session Resume.

- ClientHello message specifying previous established SessionID.



I don't know...

- premaster / master. meaning.
  - the process of making symmetric key encrypted with Server's public key. in Client
  - the process of getting the symmetric key by decrypting cipher with Server's private key in Server.

4. Chapter. Message format.

* Transport Requirements

4 different sources.
1. change Cipher (20)
2. Alert (21)
3. Hand-shake (22)
4. application (23)

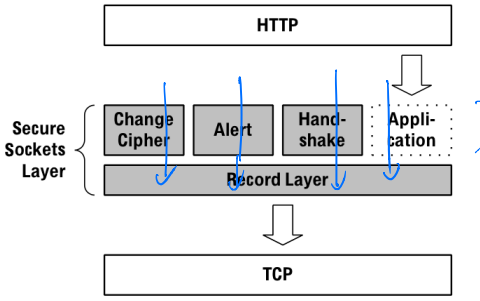


Figure 4-1 SSL consists of several component protocols.

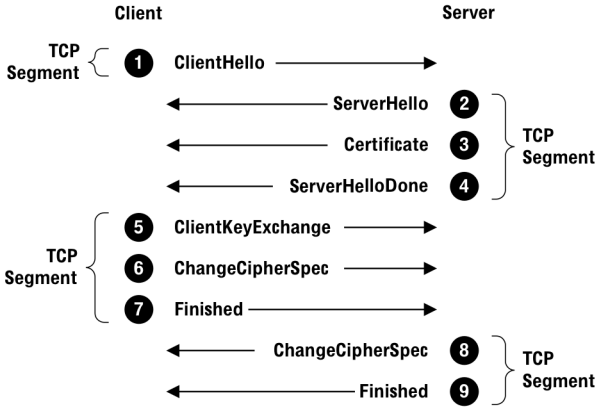
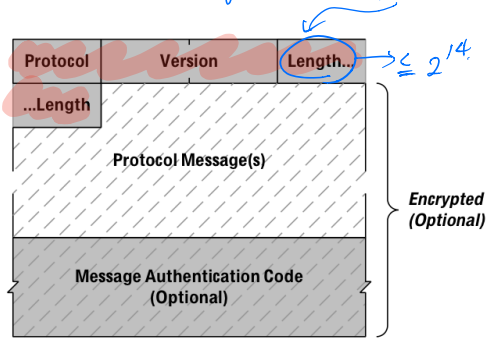


Figure 4-2 SSL can combine messages within TCP segments.

* Self-delimiting

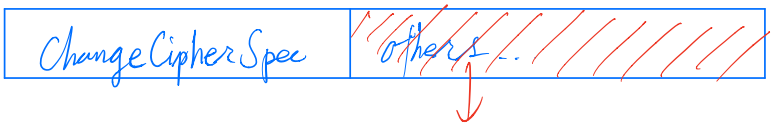
. SSL put the messages' indicator (= message length) in TCP segment. So, let SSL combine multiple SSL messages into single TCP segments.

* Record Layer (= 5bytes of meta + Protocol message + message authentication)



* 5 meta bytes help multiple higher-layer messages into a single single Record layer

* 4 different higher-layer protocols.
  - ChangeCipherSpec Protocol [.20]
    * Why this is a separate Protocol?
      → The characteristic of this ∴ It activates. Encryption after this message sent. so, if it belongs to Handshake protocol, the case of having ChangeCipherSpec and some other protocols in order has in problem for Encryption.



need to be encrypted. but. it cannot be...

Prot: 20	Vers: 3	0	Len: 0
1	CCS: 1		

→ simple ChangeCipherSpec.

- Alert Protocol

  * severity level

    1. Fatal (need to terminate session.)
    2. Warning. (could be ignored but insecure in future)

  * Error values

    0 : Close Notify
    10 : Unexpected Message.
    20 : Bad MAC.
    30 : Decompression Failure
    40 ~ 50 : Some thing handshake error

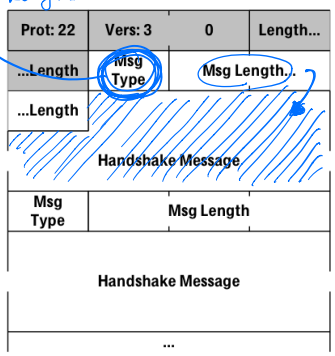
- handshake Protocol.

  * It is the one primarily responsible for negotiating SSL sessions.

Table 4-4 Handshake Protocol Types

Value	Handshake Protocol Type
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

Doesn't include message type, Length.



* HelloRequest

  - Server asks a Client to restart the SSL handshake nego.

* Client Hello

- Date : reduce message duplicating.
- session ID : previous session reuse.

☆ Don't place any information (revealed).

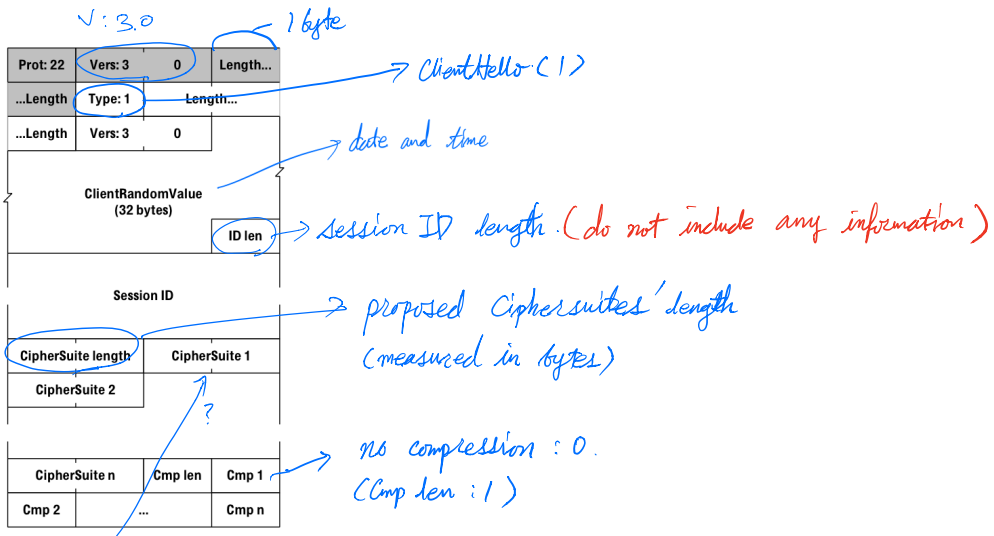
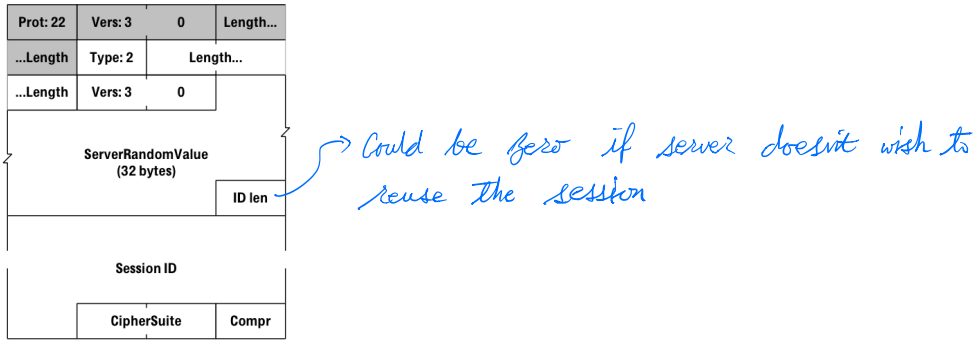


Table 4-5 SSL Version 3.0 CipherSuite Values

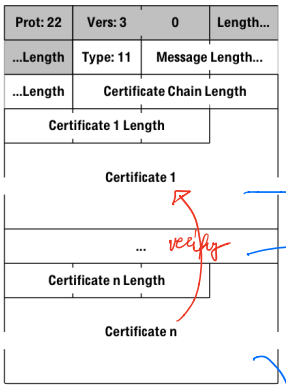
Value	Cipher Suite
0,0	SSL_NULL_WITH_NULL_NULL
0,1	SSL_RSA_WITH_NULL_MD5
0,2	SSL_RSA_WITH_NULL_SHA

* Server Hello

- Server must pick from among the choices the client proposed.



* Certificate



The order of certificates are important follows below !

1. sender's certificate
2. the authority CA that issued sender's certificate.
3. the authority CA that verifies 2's certificate

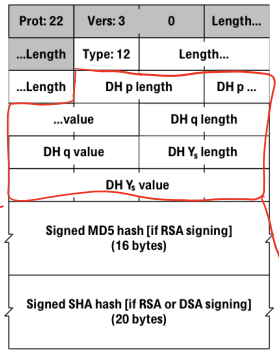
.....

until root certificate

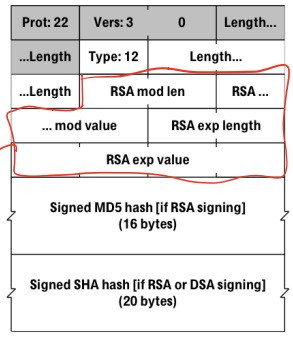
* Server Key Exchange

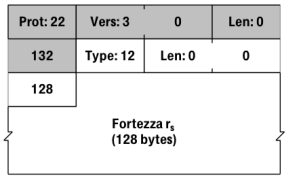
- It depends on Cryptographic Algorithm

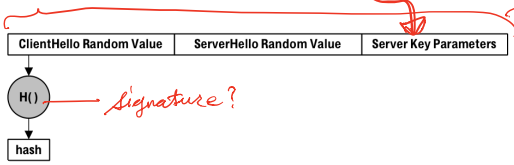
DH RSA Fortezza



* only use single signature







hash input *

signature ?

* CertificateRequest
  - to authenticate a client's identity — some information
  - ask a client to send its certificate (signed by Client's private key)
  - ask a client which certificates are acceptable

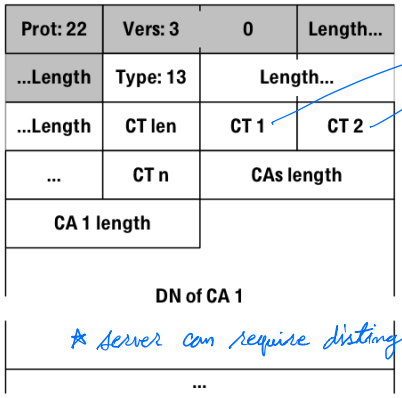


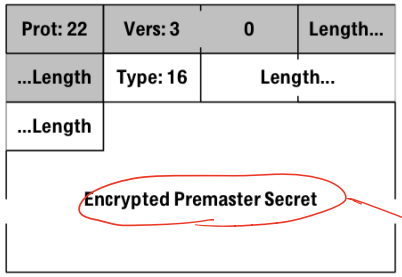
Table 4-6 Certificate Types

CT Value	Certificate Type
1	RSA signing and key exchange
2	DSA signing only
3	RSA signing with fixed Diffie-Hellman key exchange
4	DSA signing with fixed Diffie-Hellman key exchange
5	RSA signing with ephemeral Diffie-Hellman key exchange
6	DSA signing with ephemeral Diffie-Hellman exchange
20	Fortezza/DMS signing and key exchange

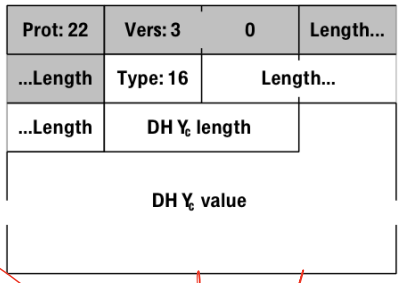
* server can require distinguish name.

* Client Key Exchange
  - send Encrypted Premaster Secret / DH Y_c Value

- RSA



- DH



ephemeral.

- Two cases of Client Key Exchange
  1. Ephemeral → enc (server pub, 2byte ver + 46 bytes random)
  2. DH exchange is explict. → Client Key Exchange will be Empty

* CertificateVerify (C → S)

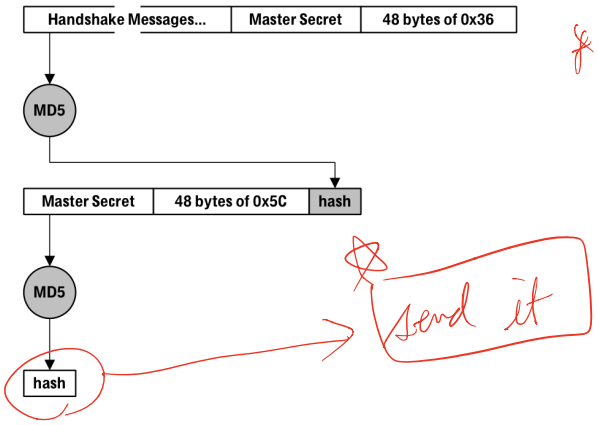
- Client prove that it possesses the private key corresponding to its public key certificate with it.
- hashed information digitally signed

  - RSA signing : MD5/SHA
  - DSA signing : SHA

?? need to update verify master secret.

```
master secret = MD5(premaster secret + SHA('A' + premaster secret +
                    ClientHello.random + ServerHello.random))
              +
                MD5(premaster secret + SHA('BB' + premaster secret +
                    ClientHello.random + ServerHello.random))
              +
                MD5(premaster secret + SHA('CCC' + premaster secret +
                    ClientHello.random + ServerHello.random))
```

* Client knows pre-master and get the master secret

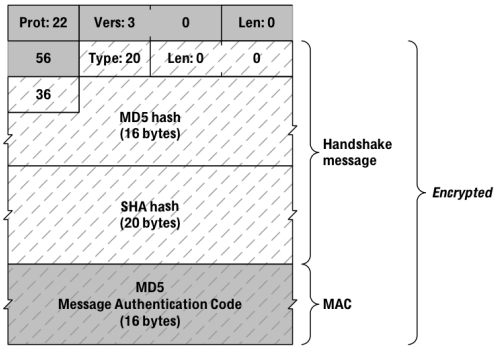


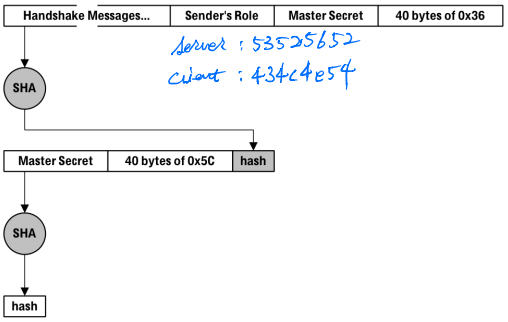
* Get the signed hash

* Finished

- type 20,
- Finished message is itself encrypted using the cipher

suite parameter





* this calculation is similar with Certificate Verify but it has sender's Role