

Asymmetric (often a.k.a. Public key) algorithms

The basis (or key ☺) of most security today!

Without PK – No SSL, No SSH, Digital signatures, email encryption, etc. etc.

Computationally **VERY** expensive

Can always in theory be broken eventually.

Public key encryption relies on the difficulty of factoring very large numbers, and testing numbers for primeness. (Or less commonly equally hard problems such as locating points on an elliptical curve).

As a result, given sufficient time, any public key algorithm can be broken, but the time required depends on key length:

But all problems that interest public key cryptographers are either P or NP.

P = can be solved in polynomial time, e.g. product of 2 numbers of n bits can be solved in at most n^2 bit operations

All problems that are P are also NP, but (it is believed) not the reverse! A problem is NP if a solution can be checked in polynomial time. Factoring a number is NP.

Diffie & Hellman published the first description of Public Key in 1976.

It had been independently discovered in the late 1960's by Ellis, Cox & Williamson at GCHQ. Some reports were declassified in 1997.

All PK algorithms are based on (presumed) Trapdoor One Way Functions. One way functions are functions that are easy to compute one way and hard the other. Trapdoor one way functions have a trapdoor that (if you know it) makes the reverse direction easy too.

Most modern cryptosystems use factoring as the “hard” problem, although Elliptic-curve algorithms hold some promise, since currently no non-exponential time solution is known.

Size of problem is directly proportional to key length.

RSA Challenges have now demonstrated the 512 bit is weak.

Increasing the key length by 1 bit doubles the effort required to brute force the key. Doubling key length therefore increases effort required by something of the order of 2^{512} .

Doubling the key length makes decrypt operations around four times slower and encrypt operations around eight times slower.

From RSA Security FAQ (www.rsasecurity.com)

	Block Cipher	RSA	Elliptic Curve	DSA
Export Grade	56	512	112	512 / 112
Traditional	80	1024	160	1024 / 160
recommendations	112	2048	224	2048 / 224
Lenstra/Verheul 2000	70	952	132	952 / 125
2010	78	1369	146 / 160	1369 / 138

Table 2. Minimal key lengths in bits for different grades.

Some key points:

Public key compromises are often very serious!

Digital signatures may be required to persist for a very long time!

What is secure today, may not be tomorrow, will not be in 100 years.

Bit length is not a good indicator of strength on its own. All things being equal, it may well be.

Paranoia is important!

Worked Example

- This is ~10 bit RSA. We know 512 bit is insecure.
- To brute force we need to test prime factors of N
- Security is therefore proportional to the difficulty of doing so!

Lets have a look at RSA:

Select 2 large (>1024 bit) prime numbers P & Q p = 17, q = 11

Compute product (N) 187

Select E where:

$$E > 1,$$

$$E < PQ$$

E and (P-1)(Q-1) have no common prime factors

We'll use E = 7

Ciphertext (C) = Message (M)^E (mod N)

We want to transmit ASCII X (88 decimal)

N & E are the public key

$$C = 88^7 \pmod{187}$$

$$C = 11$$

When this is received, it cannot be easily decrypted without knowledge of P & Q. Using these values we can compute the private key (D).

$$E * D = 1 \pmod{(p-1) * (q-1)}$$

$$7 * D = 1 \pmod{16 * 10}$$

$$D = 23$$

To decrypt we compute:

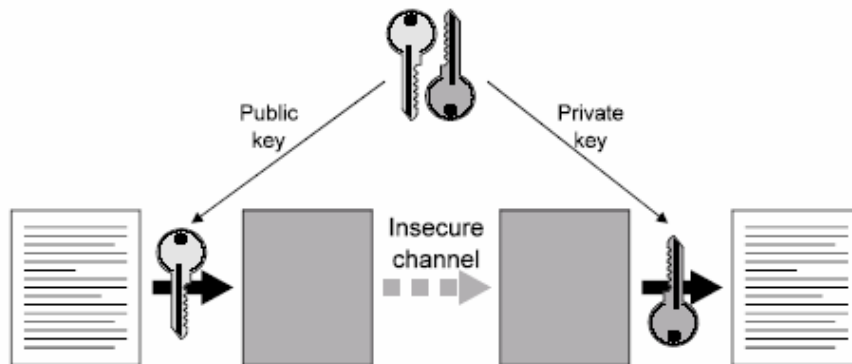
$$M = C^D \pmod{187}$$

$$M = 11^{23} \pmod{187}$$

$$M = 88$$

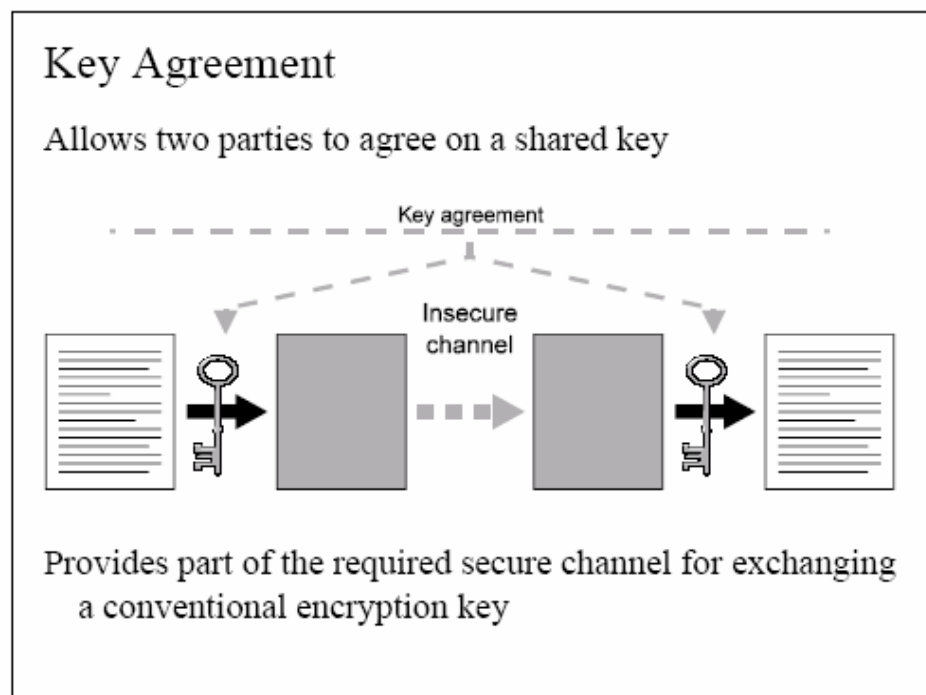
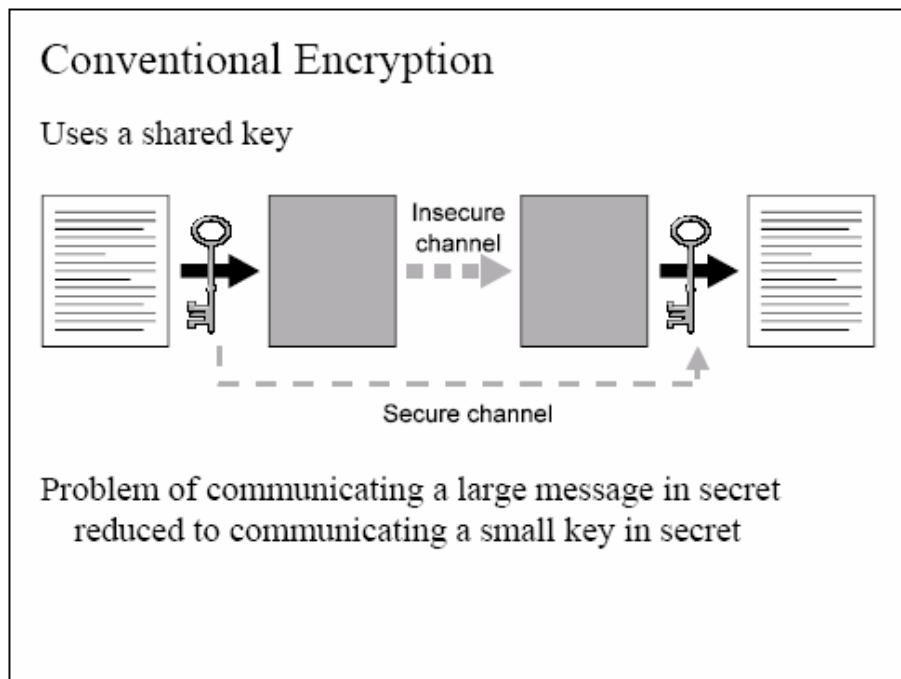
Public-key Encryption

Uses matched public/private key pairs



Anyone can encrypt with the public key, only one person can decrypt with the private key

Symmetric algorithms
Key needs to be communicated securely



Message Digests

A hash function (MD5 is a well known example)

Generates a difficult to forge “fingerprint” for a file

1 in 18,446,744,073,709,551,616

Most (current) hash algorithms are block based and iterative.

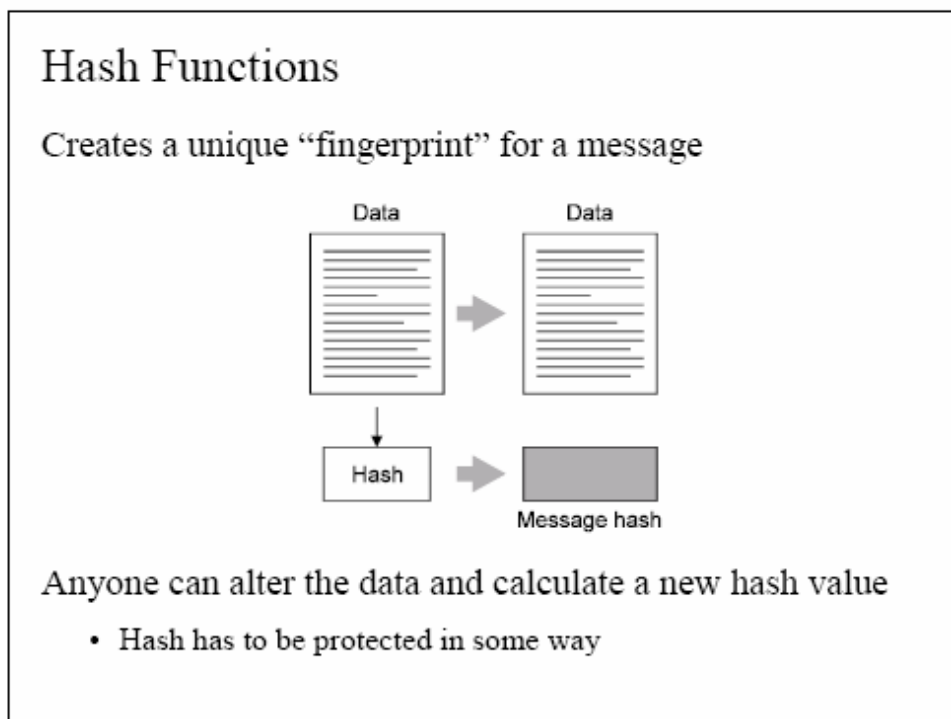
Typical block size is 512 bit.

Start with a fixed hash value H_0 .

Apply function to block $_i$, to generate H_i

Repeat until last block (which is padded out to block size)

Result is hash value.



Currently no strong hash algorithm exists!

Note: Care needs to be taken when discussing hash attacks, all hash algorithms collide, we need usefully predictable collisions for an attack.

The Birthday Attack

The Birthday Paradox:

“How many people do you need in a room to be confident two share the same birthday (excluding year)?”

probability of 50% = 23
 >99% = 60

Hash algorithms must have a probability of collision.

Taylor series expansion of the exponential function approximates:

$$n(p) \approx \sqrt{2 \cdot H \cdot \ln \left(\frac{1}{1-p} \right)},$$

H = number of outputs
p = probability required.

Solving for the birthday paradox at 50%

$$n(0.5) = 1.1774 \times \sqrt{365} = 22.494$$

MD5 = $\sim 1.8 \times 10^{19}$ unique values, we'd expect 5.1×10^9 attempts to generate a collision.

The above assumes hashes are equally probable, a best case!

The goals

- Privacy
- Authentication

In reality, any system will always have more requirements

Summary

OTPs usually are not ☺

Expertise requires research on security engineering. Let others do the crypto.

Engineering secure systems is hard. In effect we are trying to prove the non-existence of an attack under all conditions. Proving security in these situations is logically impossible.

Public Key Properties

If we know (E,N) the public key, we can encrypt so that decryption requires (D,N).

If we know (D,N) the private key, we can encrypt so that decryption requires (E,N), we can use this with a message digest as a digital signature. This allows us to authenticate.

Authentication in this case means matching a public and private key, to match to identity is harder and requires some sort of PKI infrastructure.

- What do we mean by identity?
- How does it relate to role?
- Authorisation, Authentication and Identity
- X.509 certificates common (problem with non-unique namespace)
- PGP Web of trust.
- Formal vs Informal, ad-hoc vs Hierarchical.

Other issues?

- Revocation (note: RIPA)
- Non-repudiation

Cracking Encryption

In 1998 under the direction of John Gilmore of the EFF, a team spent \$220,000 and built a machine that can go through the entire 56-bit DES key space in an average of 4.5 days. On July 17, 1998, they announced they had cracked a 56-bit key in 56 hours. The computer, called Deep Crack, uses 27 boards each containing 64 chips, and is capable of testing 90 billion keys a second.

More recently, in early 1999, Distributed.Net used the DES Cracker and a worldwide network of nearly 100,000 PCs to win the RSA DES Challenge III in a record breaking 22 hours and 15 minutes. The DES Cracker and PCs combined were testing 245 billion keys per second when the correct key was found. In addition, it has been shown that for a cost of one million dollars a dedicated hardware device can be built that can search all possible DES keys in about 3.5 hours. This just serves to illustrate that any organisation with moderate resources can break through DES with very little effort these days.

Where do we use encryption?

Encrypted Email (S/MIME, GPG, PGP, etc.)
key security?

Digital signatures (GPG, PGP)

We also use a digital signature to protect our tripwire fingerprint.

SSH – host/client verification, encrypted channel

SSL – encrypted channel, certificate based identification.

Standard use for secure HTTP

Can be used as a wrapper for any socket based service.

Supported by the openssl library.

Kerberos – encrypted network traffic

The Secure Sockets Layer (SSL)

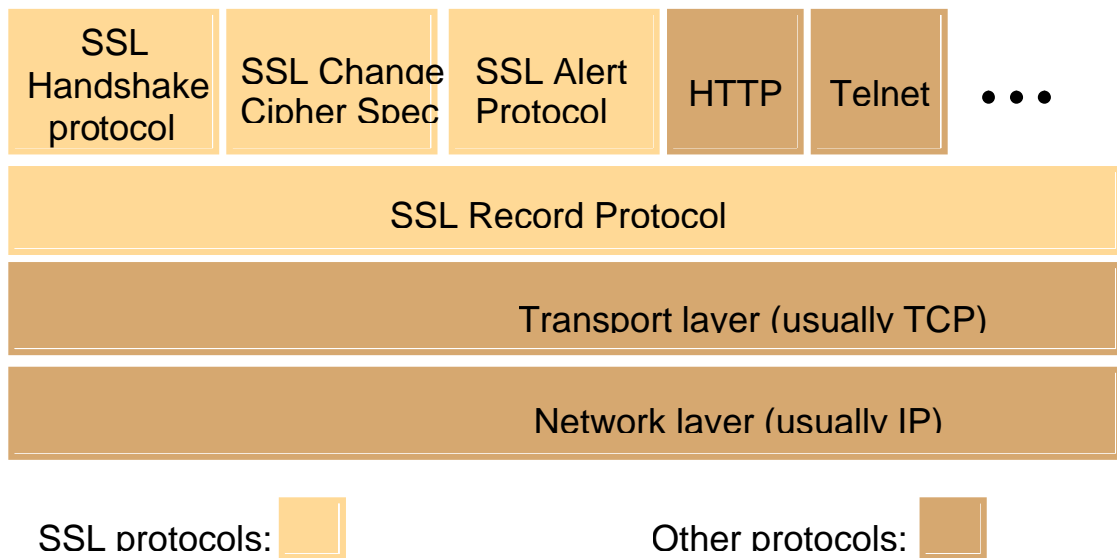
SSL is also known as TLS

Supported on most architectures

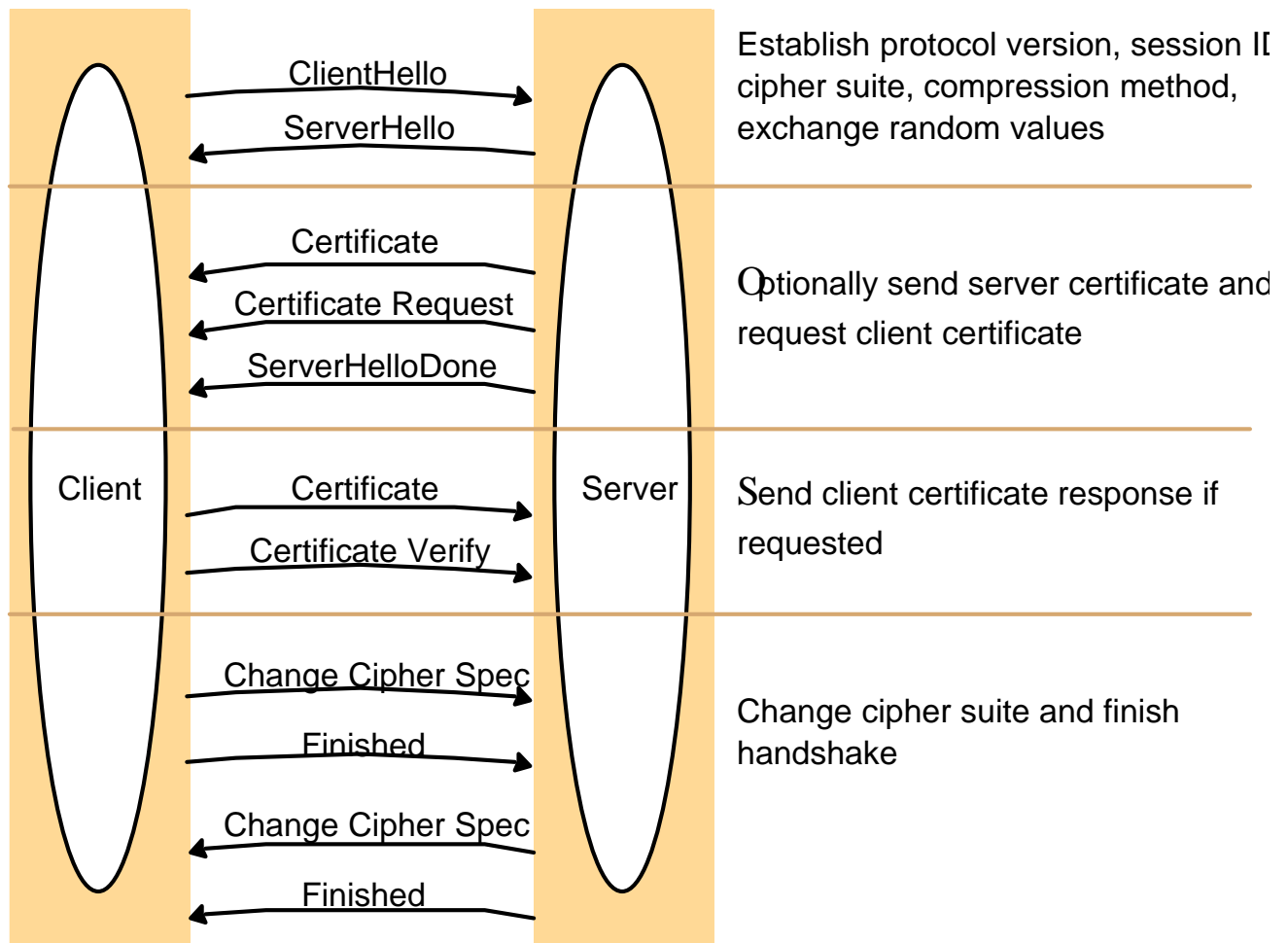
Supported by an open source library (openssl)

Two part tutorial on the module web page

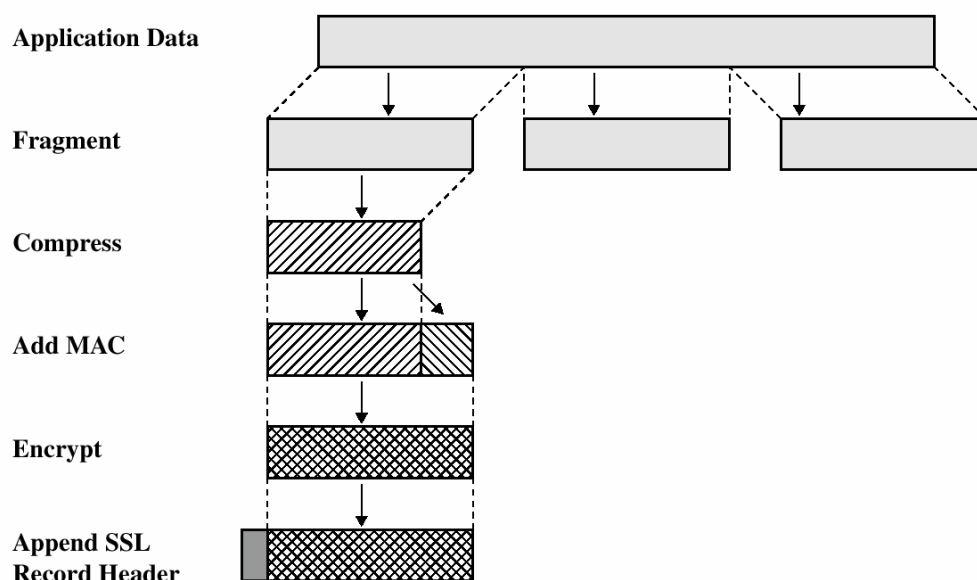
From CDK: 7.17, SSL protocol stack



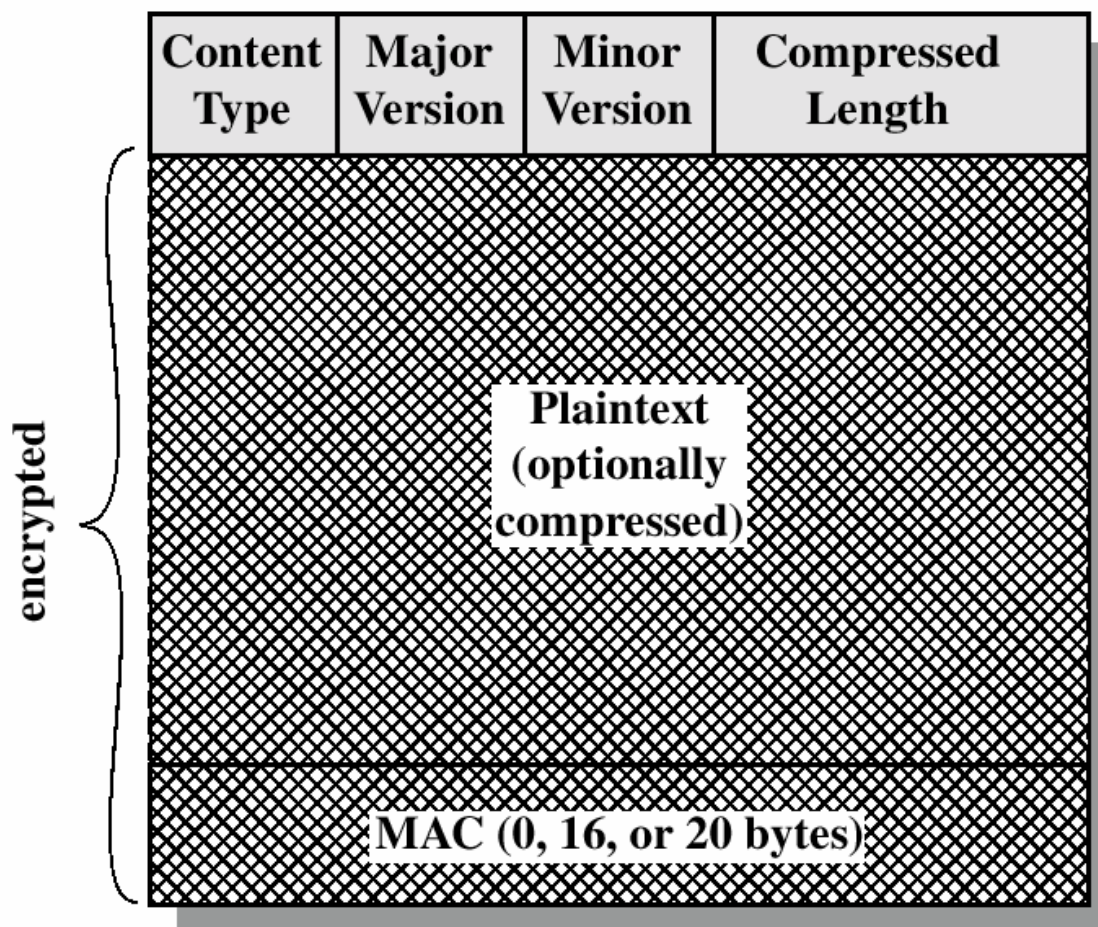
SSL Handshake (from CDK fig. 7.18)



SSL Record Protocol



SSL Record Format



SSL Handshake configuration options (CDK fig 7.19)

<i>Component</i>	<i>Description</i>	<i>Example</i>
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA
Message digest function	for creating message authentication codes	SHA

X.509 certificates

X.509 ITU standard adopted by IETF and widely used on the Internet

IETF version of X.509 is RFC 3280

X.509 is a way to describe certificates (meta- data or schema written in ASN.1)

Field	Meaning
Version	Which version of X.509
Serial number	This number plus the CA's name uniquely identifies the certificate
Signature algorithm	The algorithm used to sign the certificate
Issuer	X.500 name of the CA
Validity period	The starting and ending times of the validity period
Subject name	The entity whose key is being certified
Public key	The subject's public key and the ID of the algorithm using it
Issuer ID	An optional ID uniquely identifying the certificate's issuer
Subject ID	An optional ID uniquely identifying the certificate's subject
Extensions	Many extensions have been defined
Signature	The certificate's signature (signed by the CA's private key)