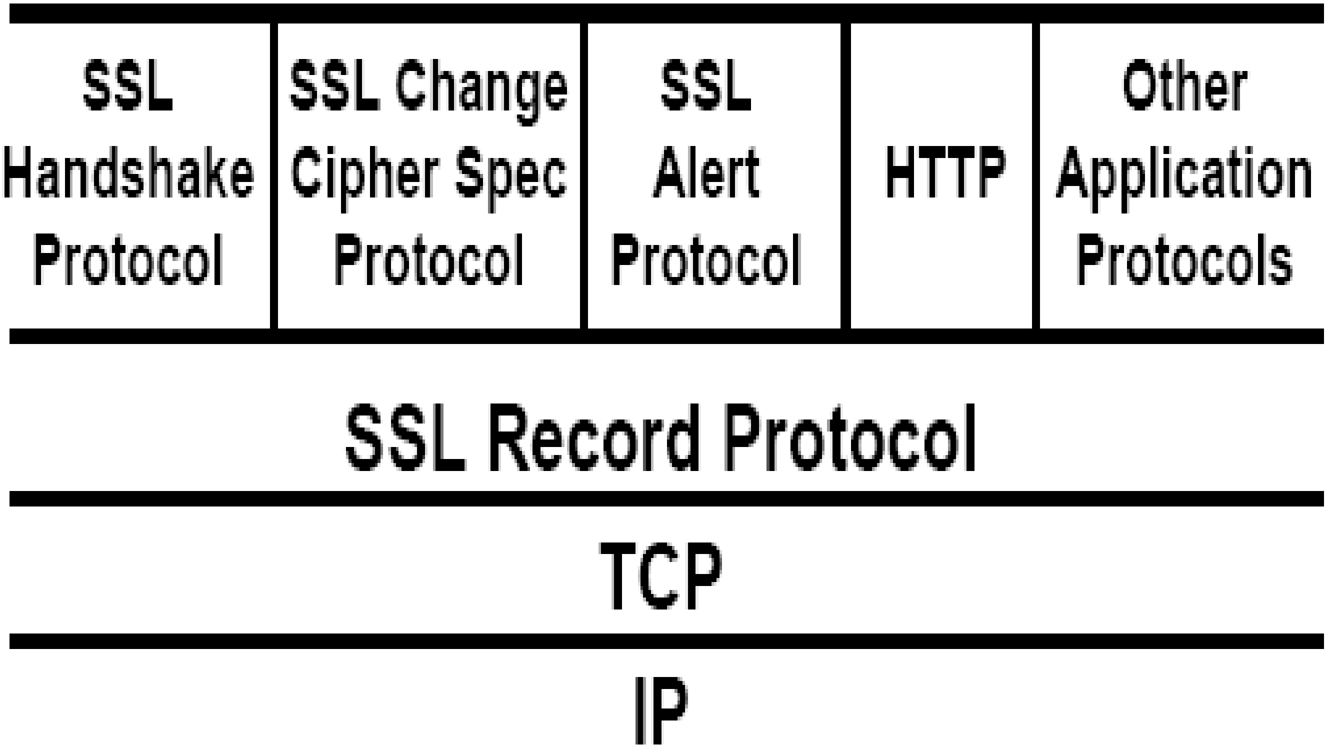
Secure Socket Layer

Position of SSL



SSL Handshake Protocol

SSL Change Cipher Spec Protocol

SSL Alert Protocol

HTTP

Other Application Protocols

SSL Record Protocol

TCP

IP

SSL ARCHITECTURE(Cont..)

- **Handshake protocol:**
  - **Establish Security Capabilites**
  - **Server & Client Authentication and key exchange**
  - **10 message types**
- **Record protocol:**
  - **fragment, compress, MAC, encrypt**
- **Alert protocol: straightforward**
  - **2 byte messages**
  - **1 byte alert level - fatal or warning; 1 byte alert code**

SSL SERVICES

- **Peer entity authentication**
- **Data confidentiality**
- **Data authentication and integrity**
- **Compression/decompression**
- **Generation/distribution of session keys**
  - **integrated into protocol**
- **Security parameter negotiation**

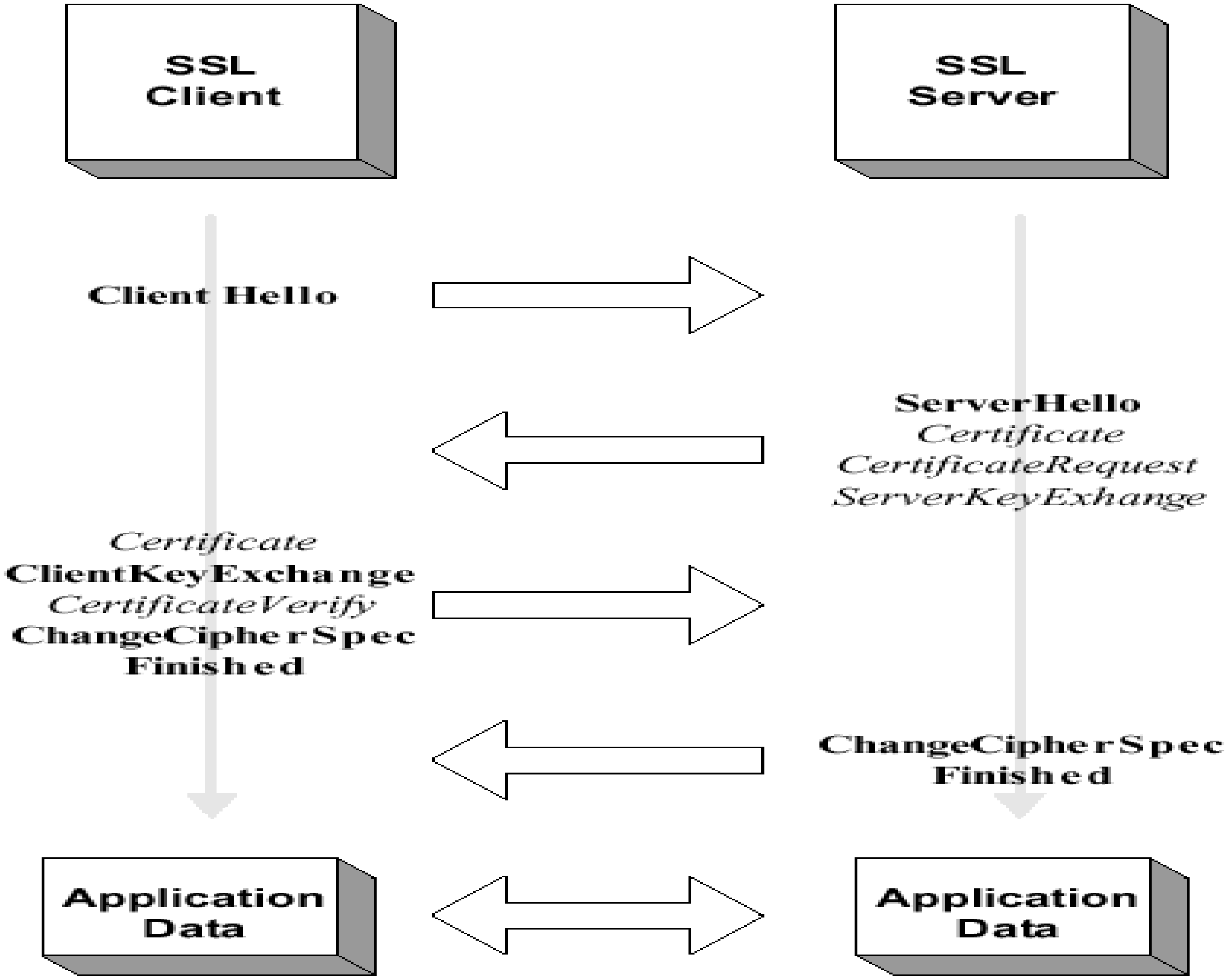
SSL HANDSHAKE PROTOCOL

- **Initially SSL session has null compression and cipher algorithms**
- **Both are set by the handshake protocol at beginning of session**
- **Handshake protocol may be repeated during the session**
- **Message Format**
    - **Type: 1 byte**
        - **10 message types defined**
    - **length: 3 bytes**
    - **content**

SSL HANDSHAKE PROTOCOL

- **Phase 1:**
  - **Establish security capabilities**
- **Phase 2:**
  - **Server authentication and key exchange**
- **Phase 3:**
  - **Client authentication and key exchange**
- **Phase 4:**
  - **Finish**

SSL HANDSHAKE PROTOCOL



SSL Client

SSL Server

Client Hello

ServerHello
Certificate
CertificateRequest
ServerKeyExhange

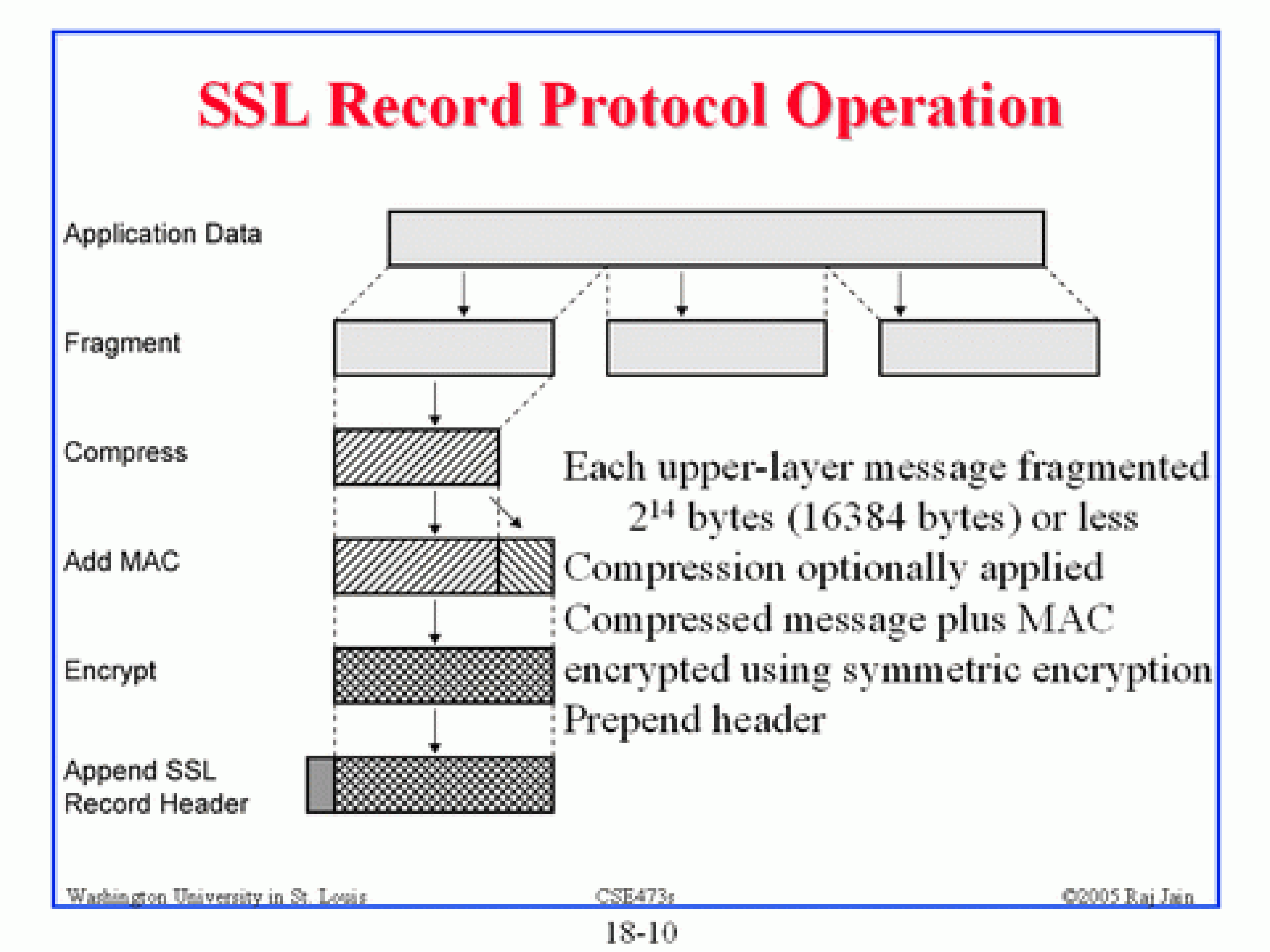
Certificate
ClientKeyExchange
CertificateVerify
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

SSL RECORD PROTOCOL



SSL Record Protocol Operation

Each upper-layer message fragmented 2^{14} bytes (16384 bytes) or less

Compression optionally applied

Compressed message plus MAC encrypted using symmetric encryption

Prepend header

SSL RECORD PROTOCOL

- **each SSL record contains**
  - **content type: 8 bits, only 4 defined**
    - **change_cipher_spec**
    - **alert**
    - **handshake**
    - **application_data**
  - **protocol version number: 8 bits major, 8 bits minor**
  - **length: max 16K bytes (actually 214 +2048)**
  - **data payload: optionally compressed and encrypted**
  - **message authentication code (MAC)**

SSL ALERT PROTOCOL

- **2 byte alert messages**
  - **1 byte level**
    - **fatal or warning**
  - **1 byte alert code**

SSL ALERT MESSAGES

- **Warning or fatal**
  - close_notify(0),
  - unexpected_message(10),
  - bad_record_mac(20),
  - decryption_failed(21),
  - record_overflow(22),
  - decompression_failure(30),
  - handshake_failure(40),
  - bad_certificate(42),
  - unsupported_certificate(43),
  - certificate_revoked(44),
  - certificate_expired(45),
  - certificate_unknown(46),
  - illegal_parameter(47),
  - unknown_ca(48),
  - access_denied(49),
  - decode_error(50),
  - decrypt_error(51),
  - export_restriction(60),
  - protocol_version(70),
  - insufficient_security(71),
  - internal_error(80),
  - user_canceled(90),
  - no_renegotiation(100

Electronic mail Security

Pretty Good Privacy (PGP)
&
S/MIME

There are two main schemes which are especially designed to provide confidentiality and authentication for electronic mail systems. These are:

PGP
(Pretty Good Privacy)

S/MIME
(Secure/Multipurpose Internet Mail Extension)

PGP

- Developed by Phil Zimmerman in 1995.
- Documentation and source code is freely available.
- The package is independent of operating system and processor.

PGP

- PGP assumes that all users are using public key cryptography and have generated a private/public key pair.
- Either RSA (with RSA digital signatures) or DSA
- All users also use a symmetric key system such as triple DES or Rijndael.

PGP services

- messages
  - authentication
  - confidentiality
  - compression
  - e-mail compatibility
  - segmentation and reassembly
- key management
  - generation, distribution, and revocation of public/private keys
  - generation and transport of session keys and IVs

PGP E-Mail Compatibility

Many electronic mail systems can only transmit blocks of ASCII text. This can cause a problem when sending encrypted data since ciphertext blocks might not correspond to ASCII characters which can be transmitted.

PGP overcomes this problem by using **radix-64 conversion**.

Radix-64 conversion

Suppose the text to be encrypted has been converted into binary using ASCII coding and encrypted to give a ciphertext stream of binary.

Radix-64 conversion maps arbitrary binary into printable characters as follows:

Radix-64 conversion

1. The binary input is split into blocks of 24 bits (3 bytes).
2. Each 24 block is then split into four sets each of 6-bits.
3. Each 6-bit set will then have a value between 0 and 2^6-1 (=63).
4. This value is encoded into a printable character.

6 bit value	Character encoding	6 bit value	Character encoding	6 bit value	Character encoding	6 bit value	Character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

What is S/MIME?

- Secure / Multipurpose Internet Mail Extension
- a security enhancement to MIME
- provides similar services to PGP
- based on technology from RSA Security
- industry standard for commercial and organizational use
- RFC 2045 to 2049

RFC 822

- defines a format for text messages to be sent using e-mail
- Internet standard
- structure of RFC 822 compliant messages
  - header lines (e.g., from: …, to: …, cc: …)
  - blank line
  - body (the text to be sent)
- example

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: “Levente Buttyan” <buttyan@hit.bme.hu>
Subject: Test
To: afriend@otherhost.bme.hu
Blablabla
```

Problems with RFC 822 and SMTP

- executable files must be converted into ASCII
  - various schemes exist (e.g., Unix UUencode)
  - a standard is needed
- text data that includes special characters (e.g., Hungarian text)

- some servers
  - reject messages over a certain size
  - delete, add, or reorder CR and LF characters
  - truncate or wrap lines longer than 76 characters
  - remove trailing white space (tabs and spaces)
  - pad lines in a message to the same length
  - convert tab characters into multiple spaces

MIME

- defines new message header fields
- defines a number of content formats (standardizing representation of multimedia contents)
- defines transfer encodings that protects the content from alteration by the mail system

MIME - New header fields

- MIME-Version
- Content-Type
  - describes the data contained in the body
  - receiving agent can pick an appropriate method to represent the content
- Content-Transfer-Encoding
  - indicates the type of the transformation that has been used to represent the body of the message
- Content-ID
- Content-Description
  - description of the object in the body of the message
  - useful when content is not readable (e.g., audio data)

MIME – Content types and subtypes

- text/plain, text/enriched
- image/jpeg, image/gif
- video/mpeg
- audio/basic
- application/postscript, application/octet-stream
- multipart/mixed, multipart/parallel, multipart/alternative, multipart/digest (each part is message/rfc822)
- message/rfc822, message/partial, message/external-body

MIME – Transfer encodings

- 7bit - short lines of ASCII characters
- 8bit - short lines of non-ASCII characters
- binary
  - non-ASCII characters
  - lines are not necessarily short
- quoted-printable
  - non-ASCII characters are converted into hexa numbers (e.g., =EF)
- base64 (radix 64)
  - 3 8-bit blocks into 4 6-bit blocks

S/MIME services

- enveloped data (application/pkcs7-mime; smime-type = enveloped-data)
  - standard digital envelop
- signed data (application/pkcs7-mime; smime-type = signed-data)
  - standard digital signature ("hash and sign")
  - content + signature is encoded using base64 encoding
- clear-signed data (multipart/signed)
  - standard digital signature
  - only the signature is encoded using base64
  - recipient without S/MIME capability can read the message but cannot verify the signature
- signed and enveloped data
  - signed and encrypted entities may be nested in any order

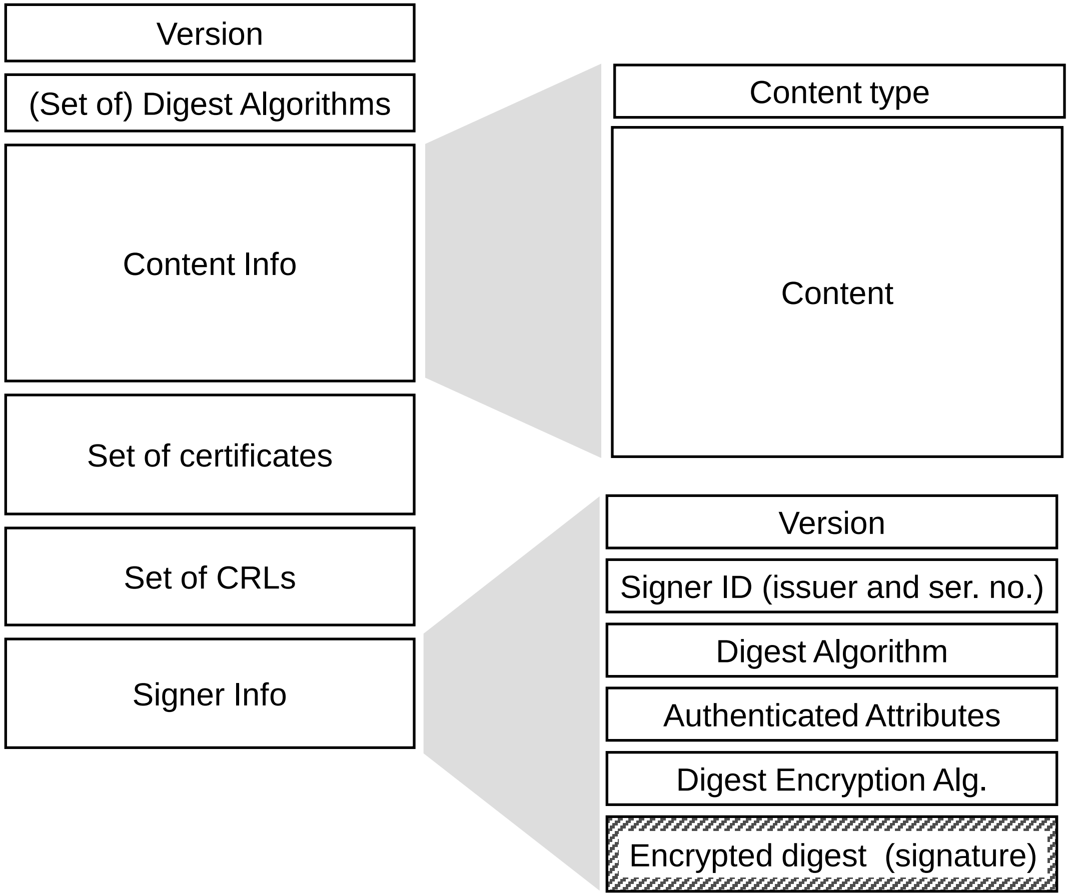
Cryptographic algorithms

- message digest
  - must: SHA-1
  - should (receiver): MD5 (backward compatibility)
- digital signature
  - must: DSS
  - should: RSA
- asymmetric-key encryption
  - must: ElGamal
  - should: RSA
- symmetric-key encryption
  - 3DES, RC2/40

Securing a MIME entity

- MIME entity is prepared according to the normal rules for MIME message preparation
- prepared MIME entity is processed by S/MIME to produce a PKCS object
- the PKCS object is treated as message content and wrapped in MIME

PKCS7 "signed data"



Version

(Set of) Digest Algorithms

Content Info

Set of certificates

Set of CRLs

Signer Info

Content type

Content

Version

Signer ID (issuer and ser. no.)

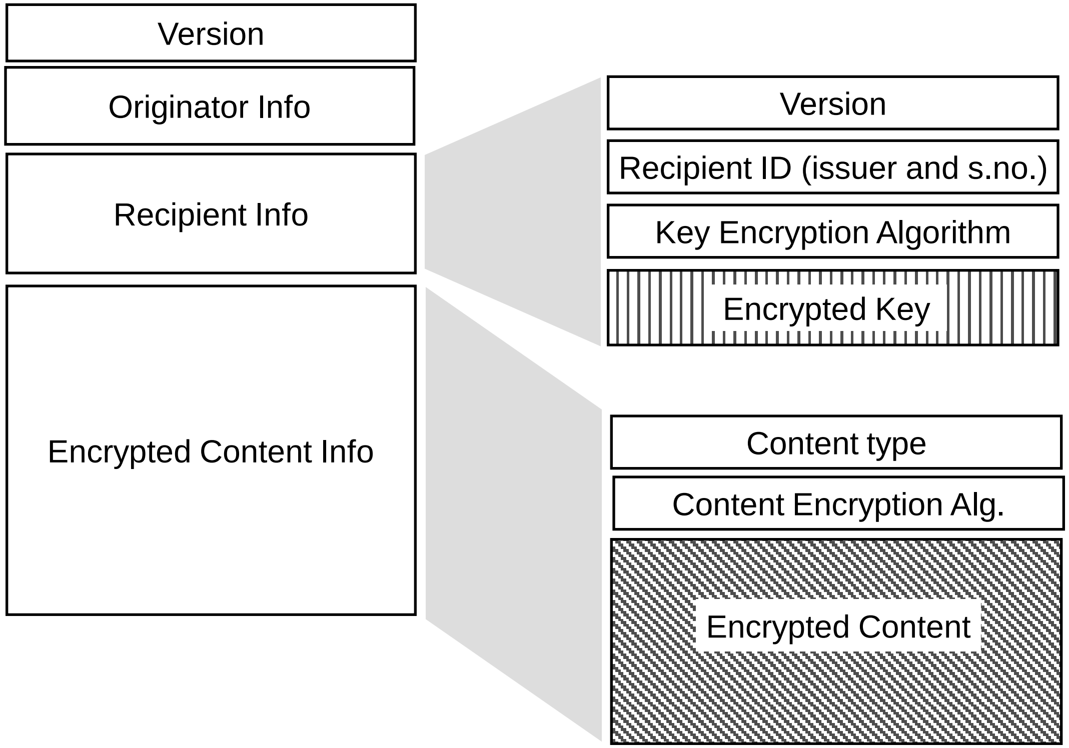
Digest Algorithm

Authenticated Attributes

Digest Encryption Alg.

Encrypted digest (signature)

PKCS7 “enveloped data”



Version

Originator Info

Recipient Info

Encrypted Content Info

Version

Recipient ID (issuer and s.no.)

Key Encryption Algorithm

Encrypted Key

Content type

Content Encryption Alg.

Encrypted Content

Enveloped data – Example

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V