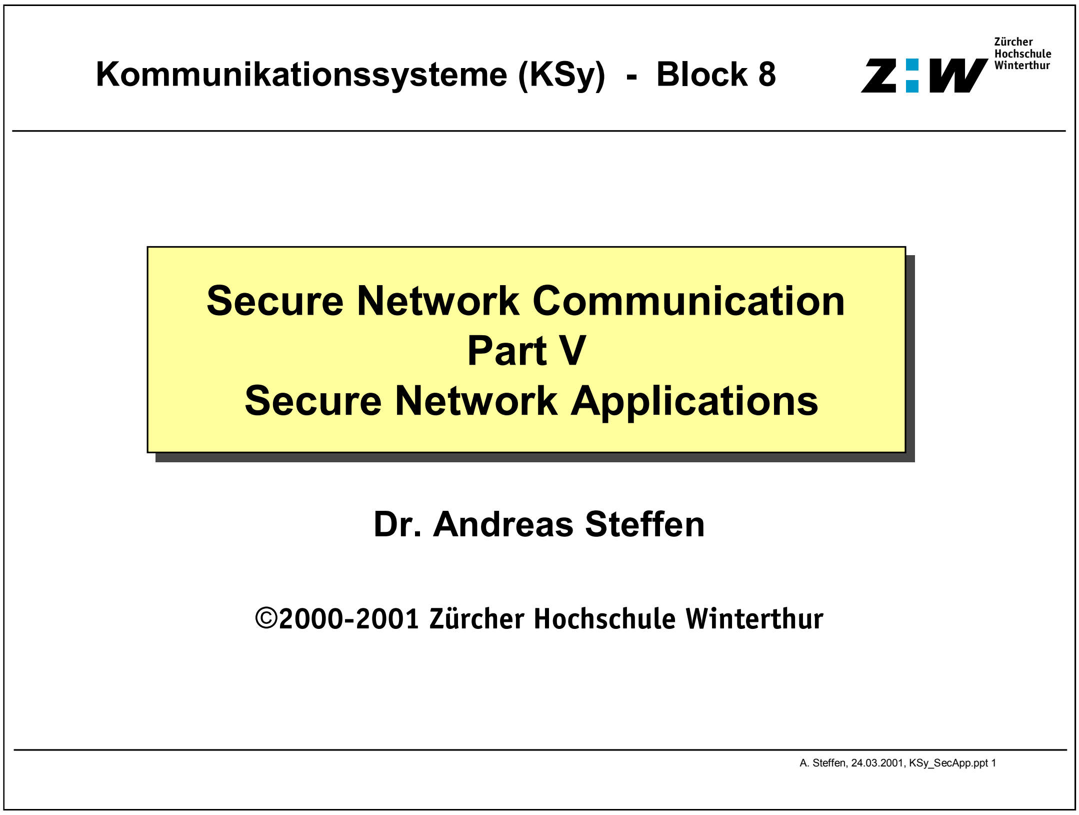


Kommunikationssysteme (KSy) - Block 8

Secure Network Communication
Part V
Secure Network Applications

Dr. Andreas Steffen

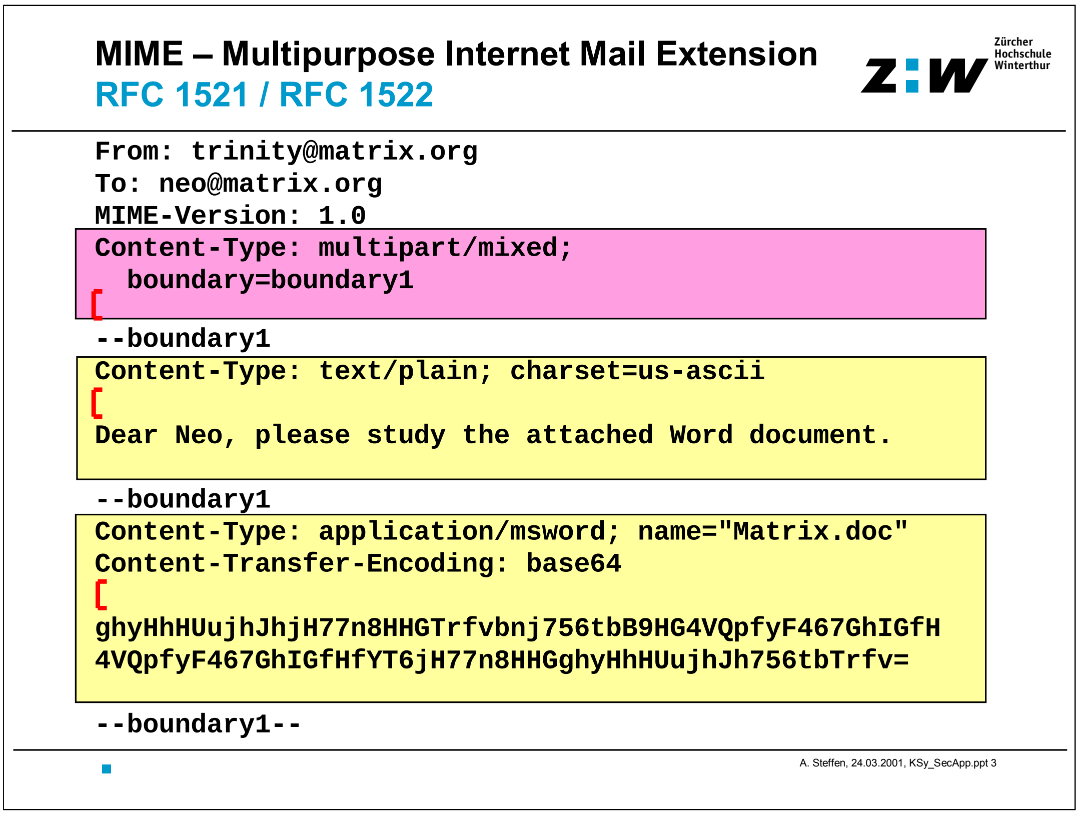
Secure E-Mail using S/MIME

- MIME – Multipurpose Internet Mail Extension
- Signed message format I (multipart/signed)
- Signed message comprising multiple attachments
- PKCS#7 – cryptographic message syntax standard
- Signed message with multiple signatures
- Signed message format II (application/pkcs7-mime with signed-data)
- Encrypted message format (application/pkcs7-mime with enveloped-data)
- Encrypted message with multiple recipients
- Signed and encrypted messages I – signing before encryption
- Signed and encrypted messages II – encryption before signing
- S/MIME configuration options for Netscape 4.7x

Secure Sockets Layer (SSL)

- SSL layer using the SSL record protocol
- SSL handshake protocol
- Implemented versions
- SSL configuration options for Netscape 4.7x and Internet Explorer 5.x
- Supported TCP-based protocols

Secure E-Mail
S/MIME



MIME – Multipurpose Internet Mail Extension
RFC 1521 / RFC 1522

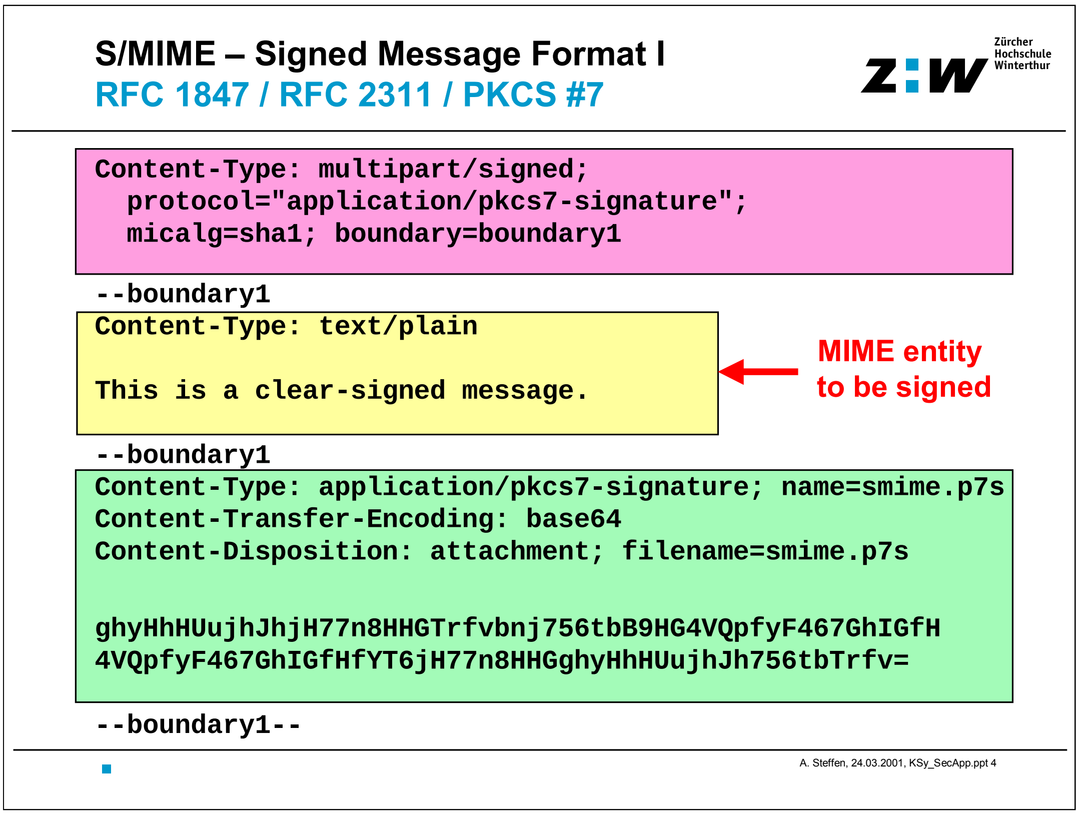
Zürcher Hochschule Winterthur

```
From: trinity@matrix.org
To: neo@matrix.org
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary=boundary1
[
--boundary1
Content-Type: text/plain; charset=us-ascii
[
Dear Neo, please study the attached Word document.

--boundary1
Content-Type: application/msword; name="Matrix.doc"
Content-Transfer-Encoding: base64
[
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=

--boundary1--
```

MIME - Multipurpose Internet Mail Extensions

- RFC 822 defines an e-mail message representation protocol which specifies considerable detail about message headers, but which leaves the message content, or message body, as **flat 7-bit ASCII text**.
- RFC 1521 redefines the format of message bodies to allow multi-part textual and non-textual message bodies to be represented and exchanged without loss of information.
- In particular, RFC 1521 is designed to provide facilities to include multiple objects in a single message, to represent body text in character sets other than US-ASCII, to represent formatted multi-font text messages, to represent non-textual material such as images, video and audio fragments, and generally to facilitate later extensions defining new types of Internet mail for use by cooperating mail agents.
- A MIME enhanced e-mail starts with a MIME header field designating the used version which currently is still **MIME-Version: 1.0**.
- The **Content-Type** header field gives the type ofa MIME attachement. Defined are the **MIME types** “text“, “multipart“, “application“, “message“, “image“, “video“, and “audio“, further subdivided into innumerous **MIME subtypes.**
- The **Content-Transfer-Encoding** header field can have the values “7bit“, “8bit“, “binary“, “base64“, “quoted-printable“ and “case-insensitive“, with “7bit“ being the default encoding.
- In the case of a multi-part entity, a "multipart" Content-Type field must appear in the entity's header. The body must then contain one or more "body parts," each preceded by an **encapsulation boundary**, and the last one followed by a **closing boundary**. Each part starts with an encapsulation boundary, and then contains a body part consisting of header area, a blank line, and a body area.

Source: RFC 1521, MIME (Multipurpose Internet Mail Extensions) Part One



S/MIME – Signed Message Format I

RFC 1847 / RFC 2311 / PKCS #7

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary1

--boundary1
Content-Type: text/plain

This is a clear-signed message.

--boundary1
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=

--boundary1--
```

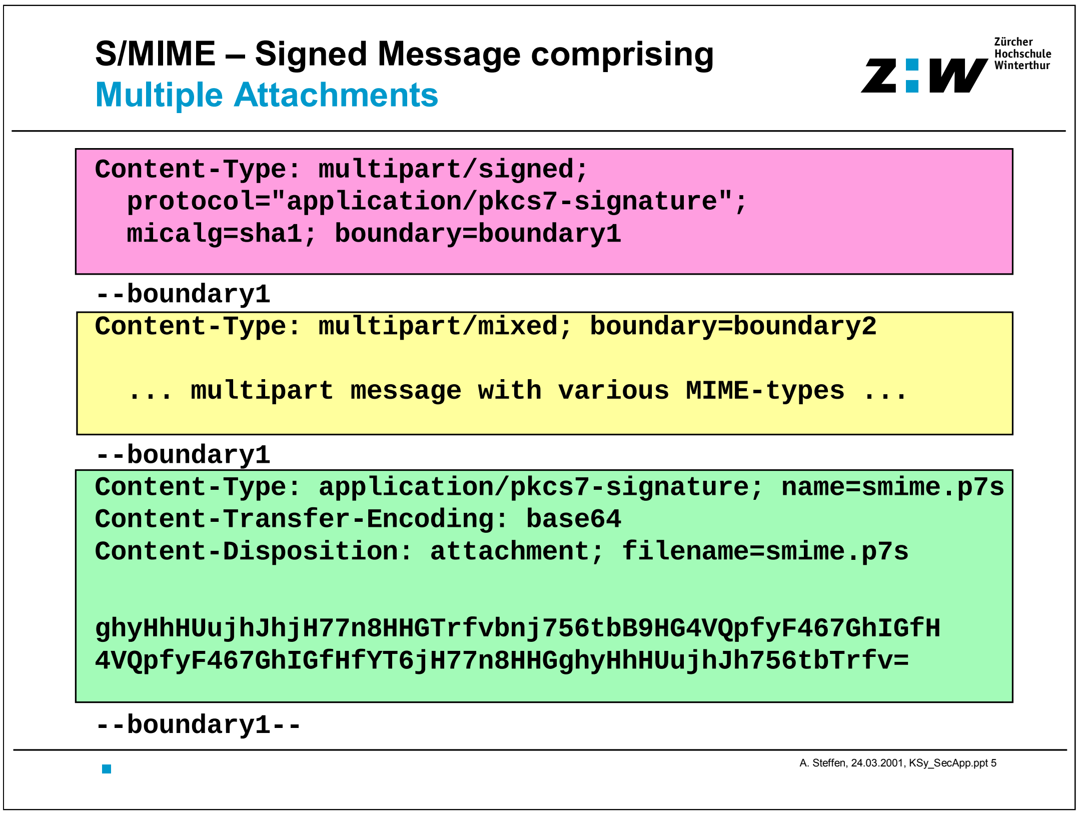
Signed Message as a MIME multi-part entity

- A signed message according to the S/MIME standard as defined by RFCs 1847 and 2311, is a multi-part entity of subtype **multipart/signed**, containing two body parts, namely

  (1) the content to be signed in regular MIME format

  (2) a **digital signature** computed over the content part (1) and contained in a binary, base64 transfer-encoded **PKCS#7 data structure** of MIME subtype **application/pkcs7-signature** or **application/x-pkcs7-signature.**

  Parts (1) and (2) are separated by a common boundary string „boundary1“.
- The multipart/signed format has the advantage that a mail client or any other e-mail application that does not support S/MIME can still read the message part (1) even if it won‘t be able to interpret the signature part (2).
- RFC 2311 recommends to add a **Content-Disposition** header field in the signature part (2), defining a filename with the extension **“*.p7s**“ which is usually set to the default filename “smime.p7s“. Older, non S/MIME-aware mail gateways might set the content-type „application/pkcs7-signature“ of the signature part to the general “application/octet-stream“ content type. The file extension defined in the content disposition field will then still allow the S/MIME client at the receiving end to recognize the binary attachment as a PKCS#7 signature.



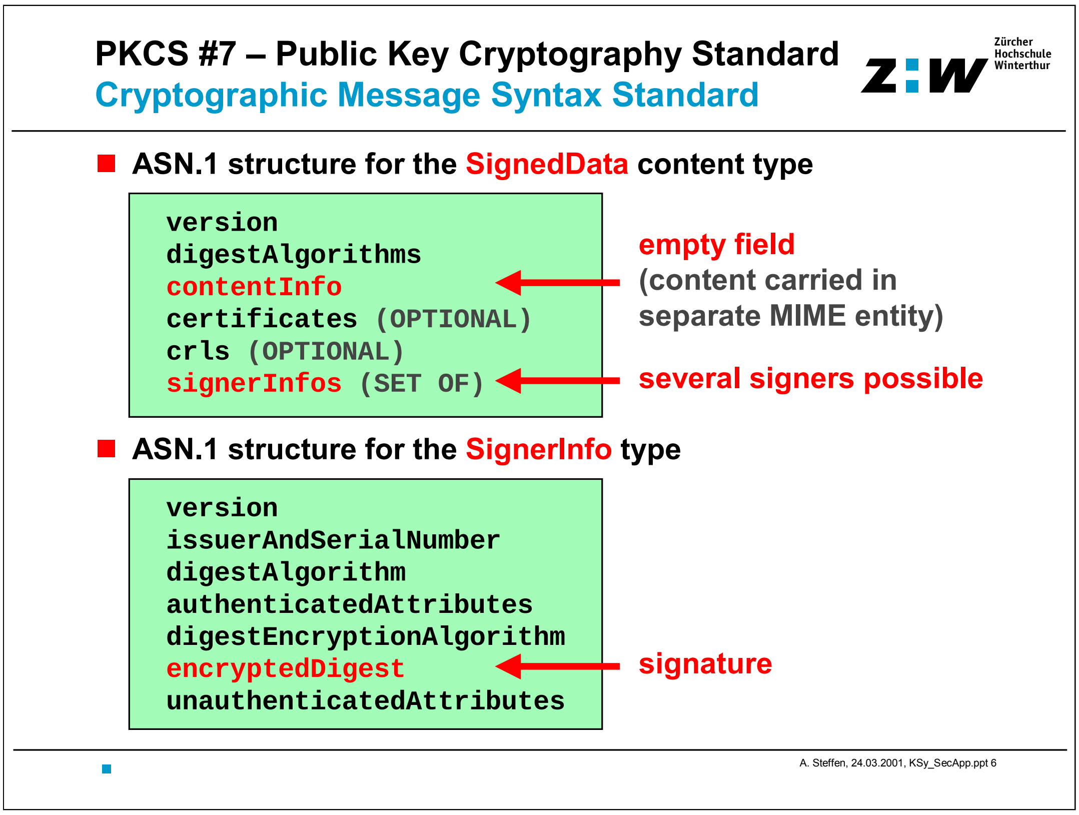
S/MIME – Signed Message comprising Multiple Attachments

Zürcher Hochschule Winterthur

```
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary1

--boundary1
Content-Type: multipart/mixed; boundary=boundary2

  ... multipart message with various MIME-types ...

--boundary1
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=

--boundary1--
```

Signature over Multi-part Messages

- Since the MIME standard explicitly supports recursively-defined multi-part entities, a signature can be computed e.g. over a **multipart/mixed** document containing several attachments, separated by a boundary string „boundary2“ that must be distinct from the boundary string „boundary1“ encapsulating the two upper-level multipart/signed body parts.
- As we will see in a moment, a S/MIME digital signature consists of a hash-value computed over the content part (1) that gets encrypted with the private key of the sender. Therefore it is of **utmost importance** that the content part (1) is not changed in any minor way by mail transfer agents along the way.
  This means that
  - MIME parts containing non US-ASCII characters like e.g. ‘ä‘, ‘ö‘, ‘ü‘ which cannot be represented by the default “ 7bit“ transfer-encoding should be transfer-encoded either as “base64“ or as “quoted-printable“, but never as “ 8bit“ or “binary“, since still not all transmission links and mail-hops support true 8-bit representations. Thus the transfer-encoding could get changed someplace under way, leading to an **invalid signature** at the receiver.
  - Signatures computed over multipart documents comprise any **MIME header fields** and **encapsulation boundaries** defined recursively within the content body part (1). So if a mail transfer agent parses the individual parts of a signed multi-part document and sends them out to the next mail hop in a different order or even exchanges the order of multiple header fields (e.g. “ Content-Transfer-Encoding “ and “ Content-Disposition “), then of course an **invalid signature** will result.



PKCS #7 – Public Key Cryptography Standard
Cryptographic Message Syntax Standard

Zürcher Hochschule Winterthur

- ASN.1 structure for the SignedData content type

```
version
digestAlgorithms
contentInfo
certificates (OPTIONAL)
crls (OPTIONAL)
signerInfos (SET OF)
```

empty field (content carried in separate MIME entity)

several signers possible

- ASN.1 structure for the SignerInfo type

```
version
issuerAndSerialNumber
digestAlgorithm
authenticatedAttributes
digestEncryptionAlgorithm
encryptedDigest
unauthenticatedAttributes
```

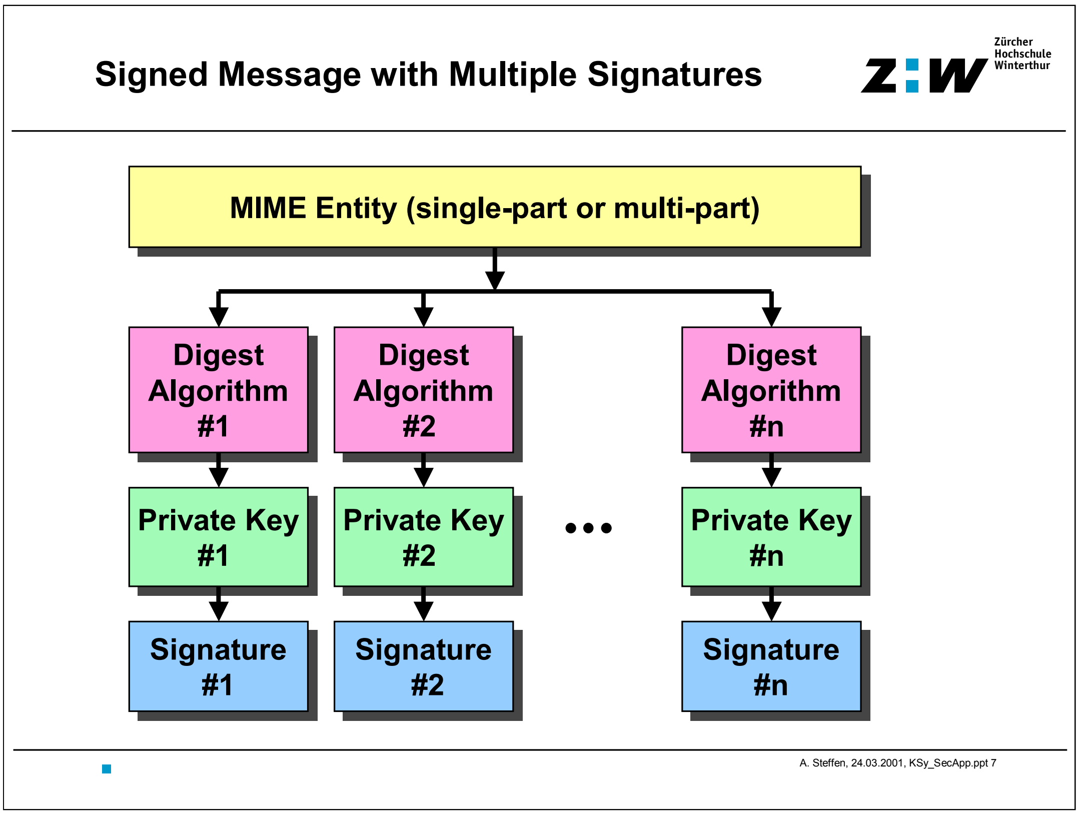
signature

PKCS #7 SignedData Object

- The **application/pkcs7-signature** part of a signed message contains a binary ASN.1 sequence object of type **SignedData.** It contains the following fields:
  - **`version`**: currently set to 1
  - **`digestAlgorithms`**: collection of message-digest algorithm identifiers (OIDs). Each element identifies the message-digest algorithm under which the content is hashed for some signer. Allows one-pass signature verification.
  - **`contentInfo`**: For the multipart/signed S/MIME format this field is empty since the content is carried in a separate MIME part. For the alternative signing format application/pkcs7-mime with parameter signed-data, the message to be signed is contained as an opaque octet string in the contentInfo field.
  - **`certificates`**: This optional field usually contains the user certificates of all the signers. Additionally it could contain certificates forming the trust chain up to the root CA(s).
  - **`crls`**: As an option any number of certificate revocation lists (CRLs) could be added.
  - **`signerInfos`**: This is a set of SignerInfo objects, containing the signer information and the digital signature of one or several signers.

PKCS#7 SignerInfo Object

- The **SignerInfo** type is an ASN.1 sequence containing the following fields
  - **`version`**: currently set to 1
  - **`issuerAndSerialNumber`**: Specifies the signer‘s certificate by issuer distinguished name and issuer-specific serial number.
  - **`digestAlgorithm`**: Identifies the message-digest under which the content and authenticated attributes (if present) are hashed.
  - **`authenticatedAttributes`**: optional authenticated attributes
  - **`digestEncryptionAlgorithm`** – specifies the encryption algorithm
  - **`encryptedDigest`**: Message digest encrypted with the signer‘s private key
  - **`unauthenticatedAttributes`**: optional unauthenticated attributes



Signed Message with Multiple Signatures

MIME Entity (single-part or multi-part)

Digest Algorithm #1

Digest Algorithm #2

Digest Algorithm #n

Private Key #1

Private Key #2

•••

Private Key #n

Signature #1

Signature #2

Signature #n

Multiple Signatures

- The **signerInfos** set structure allows to put multiple signatures on a MIME single-part or multi-part document.

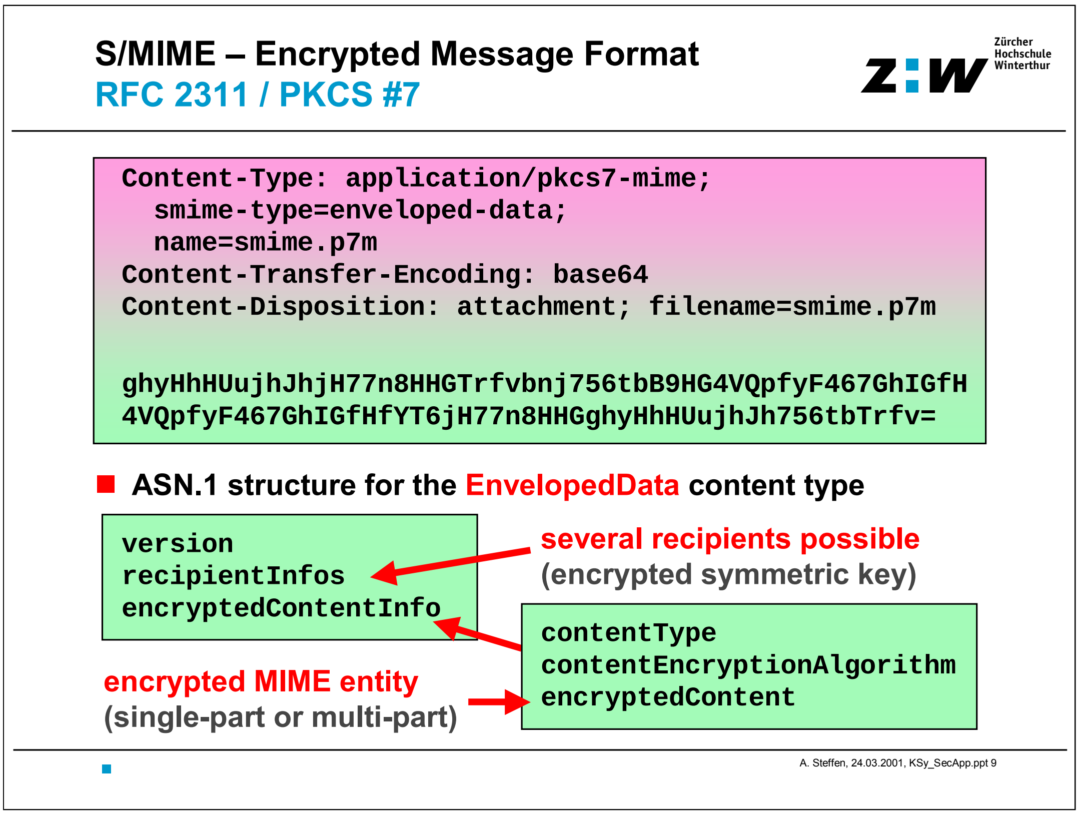
S/MIME – Signed Message Format II
RFC 2311 / PKCS #7

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

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=
```

- MIME content carried within PKCS#7 Signed Data Object
  - This alternative signing format is used e.g. by Outlook 2000
  - Pro: MIME content is not prone to changes of the transfer encoding enforced by intermediate mail transfer agents.
  - Contra: In order to read the emedded MIME message, the receiver‘s mail client must support S/MIME.

Signed Message as a MIME single-part entity

- An alternative signing format specified by RFC 2311 carries the MIME content to be signed in the opaque **ContentInfo** field of a PKCS#7 **SignedData** object.
- The PKCS#7 SignedData structure is carried in a single-part MIME entity of content type **application/pkcs7-mime** or **application/x-pkcs7-mime**.



S/MIME – Encrypted Message Format

RFC 2311 / PKCS #7

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

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfH
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbTrfv=
```

- ASN.1 structure for the EnvelopedData content type

```
version
recipientInfos
encryptedContentInfo
```

several recipients possible
(encrypted symmetric key)

```
contentType
contentEncryptionAlgorithm
encryptedContent
```

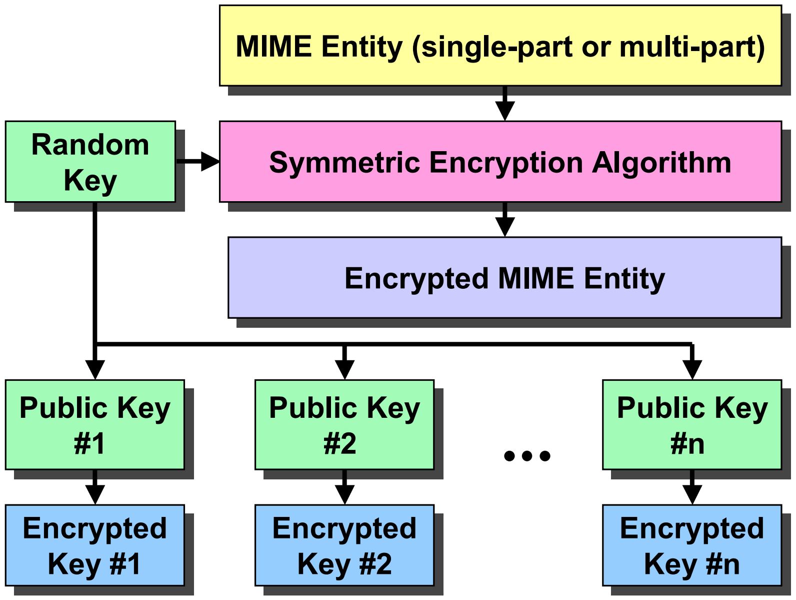
encrypted MIME entity
(single-part or multi-part)

Encrypted Messages

- Encrypted messages are transported in a single part **application/pkcs7-mime** or **application/x-pkcs-mime** MIME entity containing an ASN.1 object of type **EnvelopedData**.
- The PKCS#7 EnvelopedData object contains the fields
  - `version`: Currently set to 0
  - `recipientInfos`: Collection of per-recipient information, among other fields the symmetric session key encrypted with the recipient's public key. One or several recipients can be specified.
  - `encryptedContentInfo`: Specifies the symmetric algorithm used to encrypt the message content, plus the actual encrypted single-part or multi-part MIME entity.

Encrypted Message with Multiple Recipients

Envelope using Symmetric Encryption



MIME Entity (single-part or multi-part)

Random Key

Symmetric Encryption Algorithm

Encrypted MIME Entity

Public Key #1

Public Key #2

•••

Public Key #n

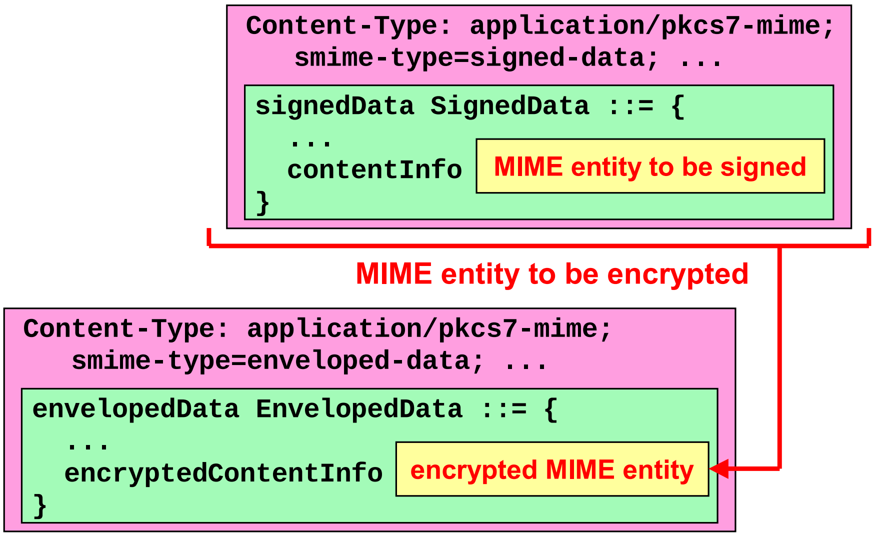
Encrypted Key #1

Encrypted Key #2

Encrypted Key #n

S/MIME – Signed and Encrypted Messages I

Signing before Encryption



```
Content-Type: application/pkcs7-mime;
    smime-type=signed-data; ...
signedData SignedData ::= {
   ...
   contentInfo
}
```

MIME entity to be signed

MIME entity to be encrypted

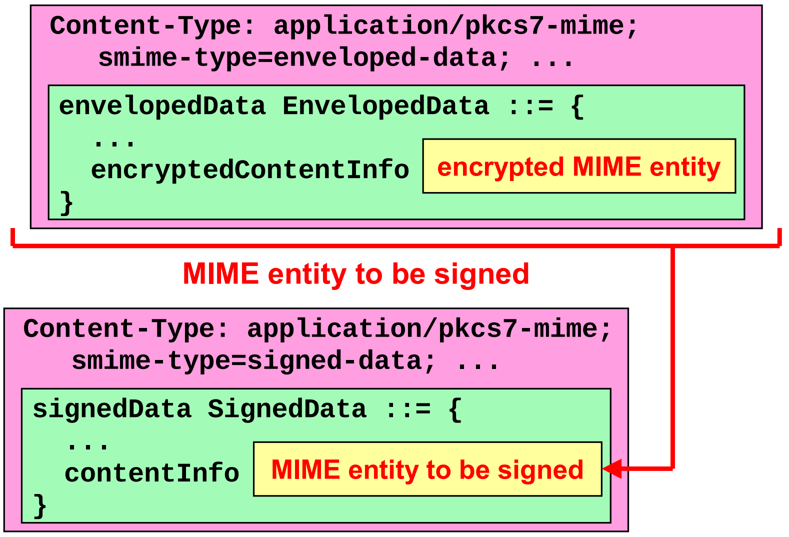
```
Content-Type: application/pkcs7-mime;
    smime-type=enveloped-data; ...
envelopedData EnvelopedData ::= {
   ...
   encryptedContentInfo
}
```

encrypted MIME entity

- Signature(s) not visible before decryption (Anonymity)

S/MIME – Signed and Encrypted Messages II

Encryption before Signing



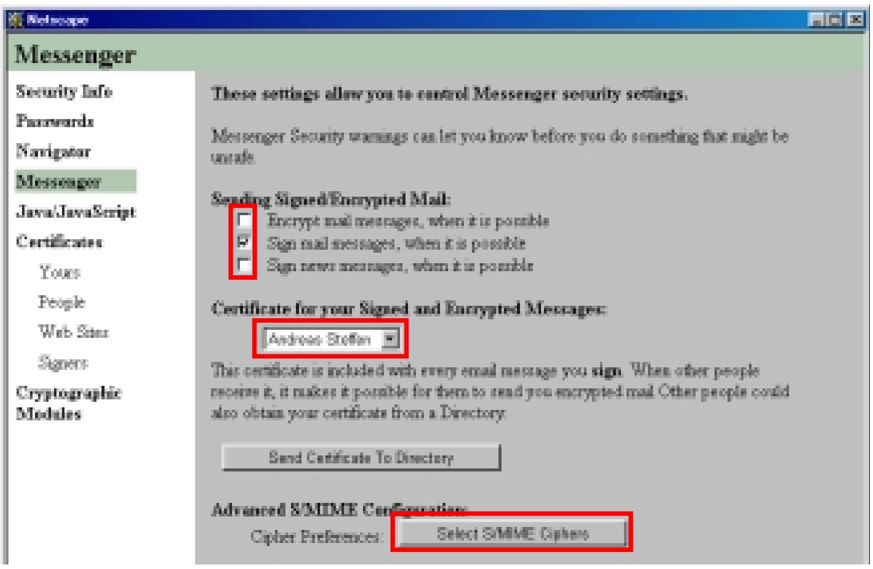
```
Content-Type: application/pkcs7-mime;
   smime-type=enveloped-data; ...
envelopedData EnvelopedData ::= {
   ...
   encryptedContentInfo  encrypted MIME entity
}
```

```
Content-Type: application/pkcs7-mime;
   smime-type=signed-data; ...
signedData SignedData ::= {
   ...
   contentInfo  MIME entity to be signed
}
```

- Signature(s) can be checked before decryption (Trust)

S/MIME - Configuration Options

Netscape 4.7x



These settings allow you to control Messenger security settings.

Messenger Security warnings can let you know before you do something that might be unsafe.

Sending Signed/Encrypted Mail:

- ☐ Encrypt mail messages, when it is possible
- ☑ Sign mail messages, when it is possible
- ☐ Sign news messages, when it is possible

Certificate for your Signed and Encrypted Messages:

Andreas Steffen

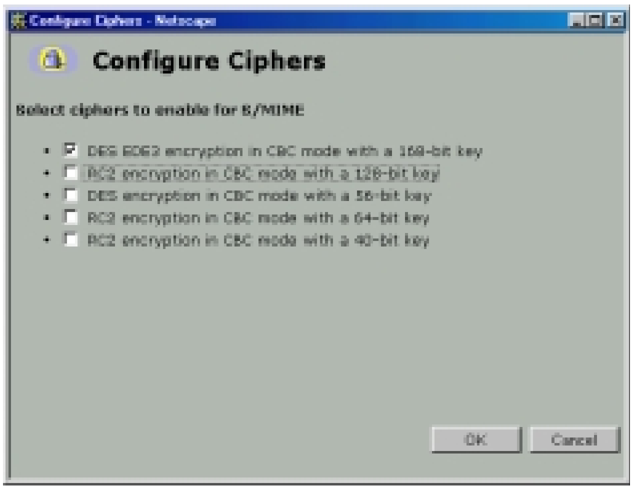
This certificate is included with every email message you **sign**. When other people receive it, it makes it possible for them to send you encrypted mail. Other people could also obtain your certificate from a Directory.

Advanced S/MIME Configuration:

Cipher Preferences:

S/MIME - Configuration Options

Netscape 4.7x

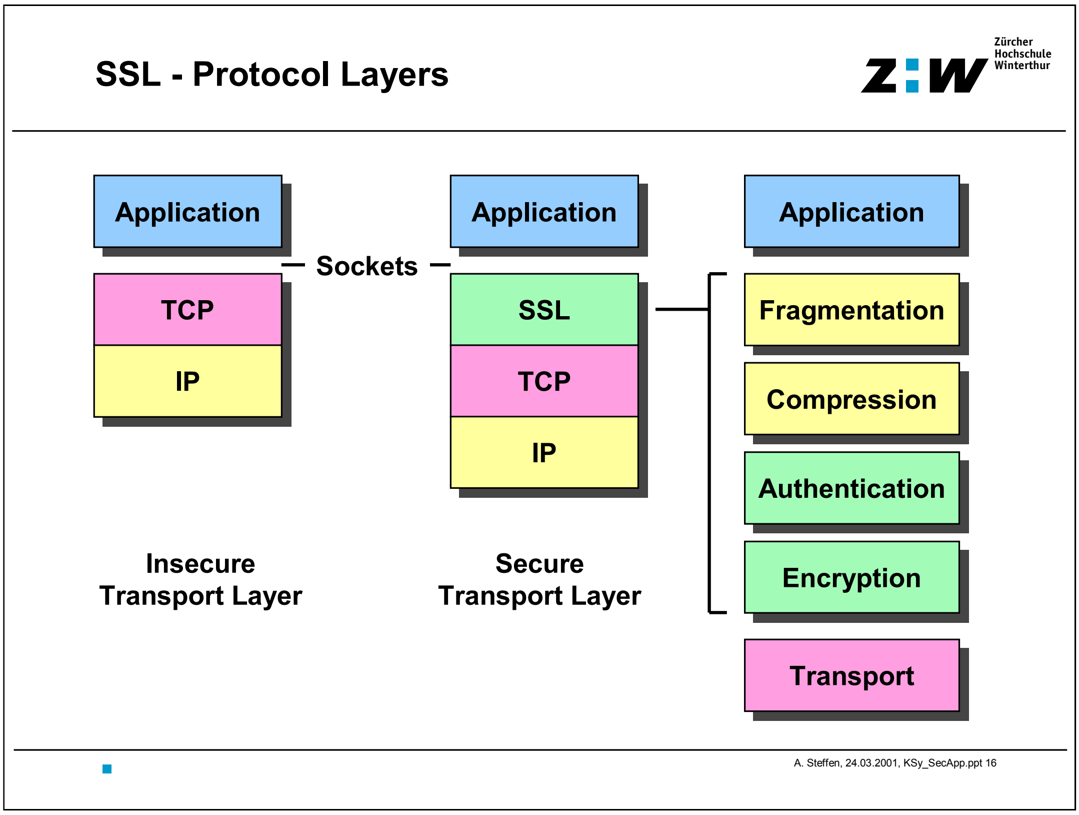


Configure Ciphers

Select ciphers to enable for S/MIME

- ☑ DES EDE3 encryption in CBC mode with a 168-bit key
- □ RC2 encryption in CBC mode with a 128-bit key
- □ DES encryption in CBC mode with a 56-bit key
- □ RC2 encryption in CBC mode with a 64-bit key
- □ RC2 encryption in CBC mode with a 40-bit key

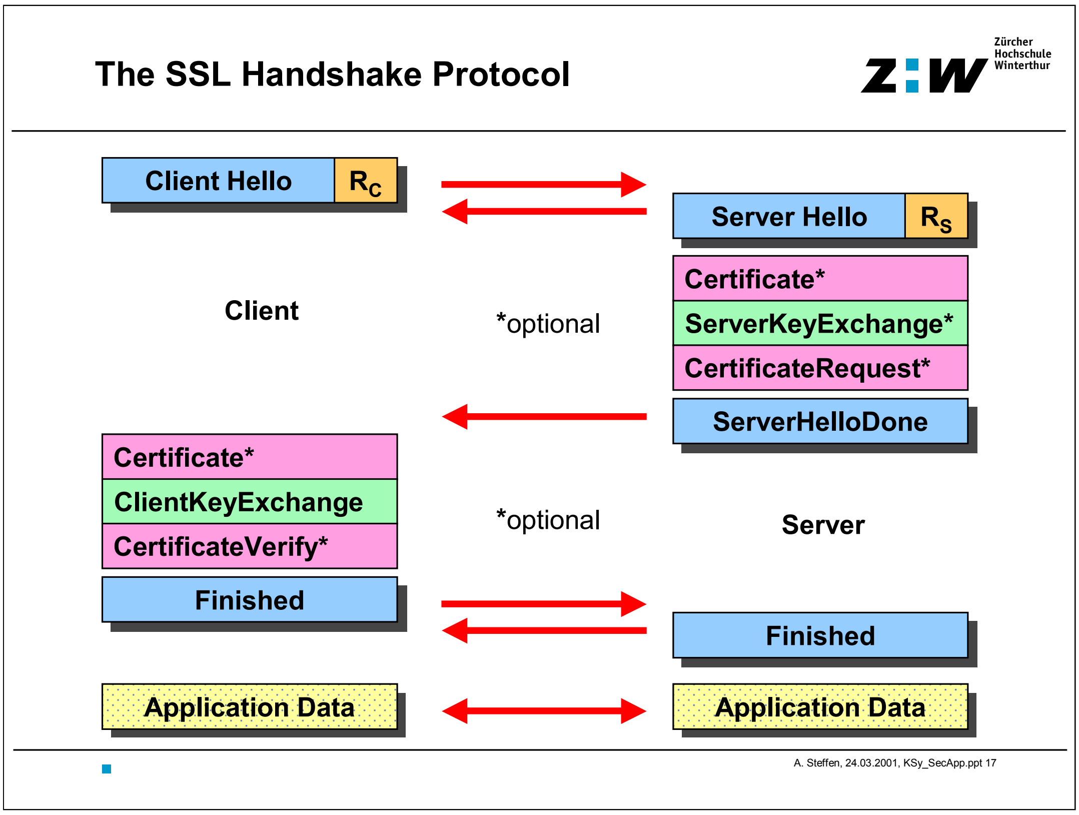
Secure Sockets Layer
SSL



SSL - Protocol Layers

SSL Protocol Layer

- The **Secure Sockets Layer** (SSL) is inserted between the **Transport layer** and the **Application Layer** (with communication layers defined according to Tanenbaum !). In contrast to **IPSec** which is a **Layer 3+** protocol based directly on IPv4 or IPv6, **SSL** is a **Layer 4+** protocol based directly on a TCP transport mechanism.
- The **SSL protocol** offers secure sockets to **SSL-aware** applications. The TCP/IP stack of the SSL client and server platforms do not have to be modified!
- The **IPSec protocol** offers secure communication to any existing IP based service or application. It is the IP stacks of the IPsec clients or IPsec security gateways that must be modified in order to support IPsec transport mode or IPSec tunnel mode, respectively.
- The SSL protocol is responsible for the following tasks:
  - Fragmentation of application data streams into SSL PDUs
  - Compression of PDUs before encryption
  - Authentication of PDUs
  - Encryption of PDUs



The SSL Handshake Protocol

Zürcher Hochschule Winterthur

Client Hello R_C

Server Hello R_S

Client

*optional

Certificate*

ServerKeyExchange*

CertificateRequest*

ServerHelloDone

Certificate*

ClientKeyExchange

CertificateVerify*

*optional

Server

Finished

Finished

Application Data

Application Data

SSL Handshake Protocol

- The SSL session state is controlled by the SSL handshake protocol that runs on top of the SSL record layer. When a SSL client and a SSL server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key encryption techniques to generate shared secrets.
- The client starts with a **ClientHello** message to which the server must respond with a **ServerHello** message – otherwise a fatal error occurs and the connection fails. The following attributes are established: Protocol Version, Session ID, Cipher Suite, and Compression Method. Additionally, two random values are generated and exchanged ClientHello-Random R_C and ServerHello-Random R_S.
- Next the server usually sends its X.509 server certificate in an optional **Certificate** message. If no certificate is sent, then an optional **ServerKeyExchange** message may be sent instead, containing the server part of a Diffie-Hellman (DH) secret. If the server insists on a **client side authentication** an optional **CertificateRequest** message is appended. The server indicates the end of the server hello phase by sending a **ServerHelloDone** message.
- If the server has sent a CertificateRequest message, the client must send either its X.509 client certificate in a **Certificate** message or a ‘no certificate‘ alert. If the client has received a server certificate containing the server‘s public RSA key, the client encrypts a randomly chosen premaster secret with it and sends it to the server in a **ClientKeyExchange** message. Alternatively the clients can send its part of a DH key exchange. Each side can now form a shared master secret.
- The client then emits a **ChangeCipherSpec** message announcing that the new parameters have been loaded, followed by a **Finished** message already encrypted with the new settings. The server does the same on its side.
- The encrypted exchange of application data can now be started.

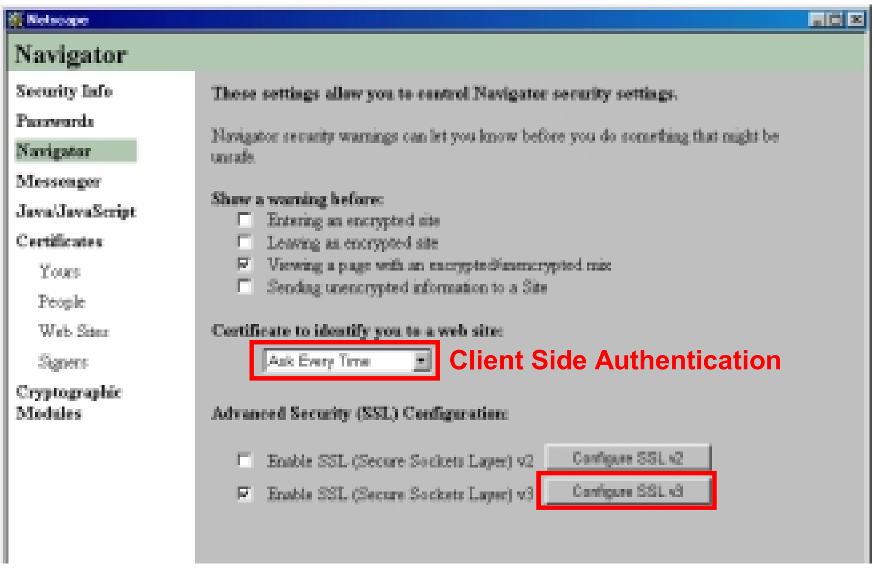
Source: Stephen Thomas, SSL and TLS Essentials, Wiley Computer Publishing

SSL – Secure Sockets Layer Protocol
Implemented Versions

- **SSL – Secure Sockets Layer Version 2.0**
  - Initially developed by Netscape
  - SSL 2.0 is sensitive to man-in-the-middle attacks leading to the negotiation of weak 40-bit encryption keys
  - Browser Support: Netscape 4.7x, Internet Explorer 5.x
- **SSL – Secure Sockets Layer Version 3.0**
  - Internet Draft authored by Netscape, November 1996
  - Browser Support: Netscape 4.7x, Internet Explorer 5.x
- **TLS – Transport Layer Security Version 1.0**
  - IETF RFC 2246, January 1999
  - TLS 1.0 ist not backwards compatible to SSL 3.0
  - Browser Support: Internet Explorer 5.x

SSL - Configuration Options

Netscape 4.7x



Netscape

Navigator

These settings allow you to control Navigator security settings.

Navigator security warnings can let you know before you do something that might be unsafe.

Show a warning before:

- □ Entering an encrypted site
- □ Leaving an encrypted site
- ☑ Viewing a page with an encrypted/unencrypted mix
- □ Sending unencrypted information to a Site

Certificate to identify you to a web site: Ask Every Time

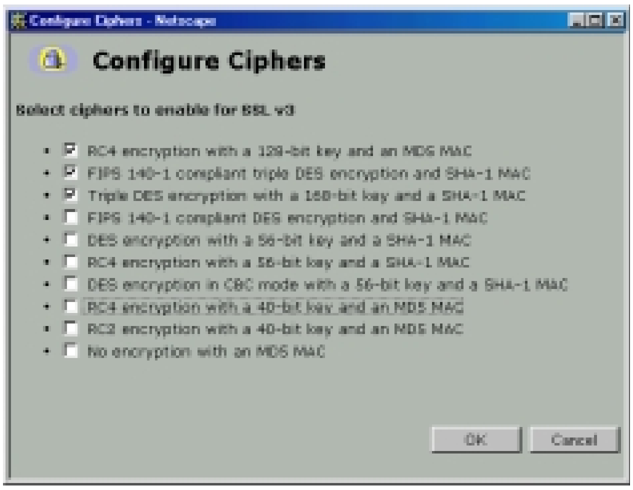
Client Side Authentication

Advanced Security (SSL) Configuration:

- □ Enable SSL (Secure Sockets Layer) v2 — Configure SSL v2
- ☑ Enable SSL (Secure Sockets Layer) v3 — Configure SSL v3

SSL - Configuration Options

Netscape 4.7x

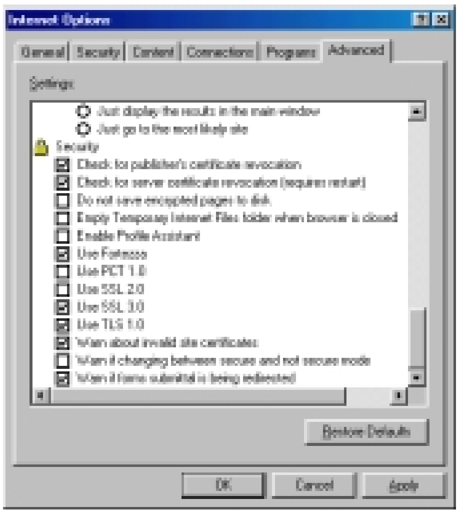


Configure Ciphers

Select ciphers to enable for SSL v3

- ☑ RC4 encryption with a 128-bit key and an MD5 MAC
- ☑ FIPS 140-1 compliant triple DES encryption and SHA-1 MAC
- ☑ Triple DES encryption with a 168-bit key and a SHA-1 MAC
- ☐ FIPS 140-1 compliant DES encryption and SHA-1 MAC
- ☐ DES encryption with a 56-bit key and a SHA-1 MAC
- ☐ RC4 encryption with a 56-bit key and a SHA-1 MAC
- ☐ DES encryption in CBC mode with a 56-bit key and a SHA-1 MAC
- ☐ RC4 encryption with a 40-bit key and an MD5 MAC
- ☐ RC2 encryption with a 40-bit key and an MD5 MAC
- ☐ No encryption with an MD5 MAC

SSL - Configuration Options
Internet Explorer 5.x



Settings:

- ○ Just display the results in the main window
- ○ Just go to the most likely site

Security

- ☑ Check for publisher's certificate revocation
- ☑ Check for server certificate revocation (requires restart)
- □ Do not save encrypted pages to disk
- □ Empty Temporary Internet Files folder when browser is closed
- □ Enable Profile Assistant
- ☑ Use Fortezza
- □ Use PCT 1.0
- □ Use SSL 2.0
- ☑ Use SSL 3.0
- ☑ Use TLS 1.0
- ☑ Warn about invalid site certificates
- □ Warn if changing between secure and not secure mode
- ☑ Warn if forms submittal is being redirected

SSL – Supported TCP-based Protocols

Service Name	Port	Secured Service
https	443/tcp	http protocol over TLS/SSL
smtps	465/tcp	smtp protocol over TLS/SSL
nntps	563/tcp	nntp protocol over TLS/SSL
sshell	614/tcp	SSLshell
ldaps	636/tcp	ldap protocol over TLS/SSL
ftps-data	989/tcp	ftp protocol, data, over TLS/SSL
ftps	990/tcp	ftp, control, over TLS/SSL
telnets	992/tcp	telnet protocol over TLS/SSL
imaps	993/tcp	imap4 protocol over TLS/SSL
ircs	994/tcp	irc protocol over TLS/SSL
pop3s	995/tcp	pop3 protocol over TLS/SSL