|  |  |  |
| --- | --- | --- |
| INTERNATIONAL TELECOMMUNICATION UNION | | **STUDY GROUP 17** |
| **TELECOMMUNICATION STANDARDIZATION SECTOR**  STUDY PERIOD 2013-2016 | | TD 1259 |
| **English only**  **Original: English** |
| **Question(s):** | 11/17 | Geneva, 17 – 26 September 2014 |
| **TD** | | |
| **Source:** | Editors | |
| **Title:** | X.cmail Certified Mail Transport and Certified Post Office Protocols | |

X.cmail

Certified Mail Transport and Certified Post Office Protocols

Summary

The objective of CMTP (Certified Mail Transfer Protocol) and CPOP (Certified Post Office Protocol) is to foster the exchanges of electronic certified mails in the world in a secure way to provide confidentiality, identification of the correspondents, integrity and non-repudiation.

Introduction

<Optional - This clause should appear only if it contains information different from Scope and Summary>

# 1 Scope

CMTP/CPOP enable to:

– Solve repudiation issues thanks to the use of electronic signature,

– Solve confidential issues thanks to the use of encryption.

– Produce reliable notices of deposit, notices of transit and notices reception,

– Use a Cmail server to track the certified mails and avoid them to get lost,

– Use a TLS transport connection to provide stronger identification. This stronger level of identification is required by the Cmail server.

The intent of this Recommendation is to extend the capabilities of SMTP and POP3 to support identification, security and not least non-repudiation to make e-mails legally bindings.

To do so, two new protocols are defined:

– Certified Mail Transfer Protocol (CMTP)

– Certified Post Office Protocol (CPOP)

The principle of these protocols is to add a set of new security commands and to update some of the existing ones within SMTP and POP3. These commands (new ones and upgrades) would enable to:

– Produce a notice of deposit electronically signed by the sender,

– Produce a notice of reception electronically signed by the recipient,

– Produce a notice or transit signed by the Cmail server

– Encrypt the message end-to-end using the recipient public key and using TLS for encryption transport between entities.

CMTP and CPOP also introduce the concept of a Cmail server which role is to certify that the exchange between two parties has indeed occurred.

The objective of CMTP and CPOP is to exchange electronic mails that would enable to certify the deposit and the reception of the email.

CMTP and CPOP are depending on an existing public-key infrastructure, although there are specific requirements on the structure of public-key certificates. There are different types of public-key certificates for Cmail servers and for Cmail clients.

# 2 References

[ITU-T X.509] Recommendation ITU-T X.509 (2012) | ISO/IEC 9594-8:2014, *Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*.

[IETF RFC 1969] IETF 1969 (1996), *Post Office Protocol – Version 3*

[IETF RFC 5321] IETF 5321 (2008), *Simple Mail Transfer Protocol*

[IETF RFC 5246]IETF 5246 (2008), *Transport Layer Security – Version 1.2*

ASN.1 Recommendation ITU-T X.680 (2008) | ISO/IEC 8824-1:2008

XML W3C XML1.0 2000 Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation

XSD W3C XML Schema 2001 XML Schema part 1: Structure, W3C Recommendation

# 3 Definitions

## 3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

**3.1.1** **certificate validation** [ITU-T X.509]: The process of ensuring that a certificate was valid at a given time, including possibly the construction and processing of a certification path, and ensuring that all certificates in that path were valid (i.e., were not expired or revoked) at that given time.

**3.1.2** **certification authority (CA)** [ITU-T X.509]: An authority trusted by one or more users to create and assign public-key certificates. Optionally the certification authority may create the subjects' keys.

**3.1.3** **hash function** [ITU-T X.509]: A (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. A "good" hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range.

**3.1.4 private key** [ITU-T X.509]: (In a public key cryptosystem) that key of an entity's key pair which is known only by that entity.

**3.1.5** **public key** [ITU-T X.509]: (In a public key cryptosystem) that key of a user's key pair which is publicly known.

**3.1.6 public-key certificate (PKC)** [ITU-T X.509]**:** The public key of a user, together with some other information, rendered unforgeable by digital signature with the private key of the CA which issued it.

**3.1.7 public-key infrastructure (PKI)** [ITU-T X.509]**:** The infrastructure able to support the management of public keys able to support authentication, encryption, integrity or non-repudiation services.

3.1.8 **Simple Mail Transfer Protocol (SMTP)** [IETF RFC 1969]: TCP/IP protocol used to send email.

3.1.9 **Post Office Protocol (POP3)** [IETF RFC 5321]: TCP/IP protocol used to receive email.

## 3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

**3.2.1** **Certified Mail Transfer Protocol (CMTP):** TCP/IP protocol used to send Certified Mail.

**3.2.2 Certified Pop Office Protocol (CPOP):** TCP/IP protocol used to receive Certified Mail.

**3.2.3 Certified Mail:** Electronic mail exchanged using CMTP and CPOP.

**3.2.4 Notice of deposit:** Electronic document signed by the sender and the Cmail server, containing information allowing to certify that a certified mail deposit occurred.

**3.2.5 Notice of reception:** Electronic document signed by the recipient and the Cmail server, containing information allowing to certify that a certified mail was received by the recipient.

**3.2.6** **Notice of transit:** Electronic document signed by Cmail servers involved in the transaction and containing information allowing to certify that the certified mail was transmitted to the Cmail server.

**3.2.7 Cmail server:** Legal entity involved in certified mail transaction.

# 4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

ASN.1 Abstract Syntax Notation One

CMTP Certified Mail Transfer Protocol

CPOP Certified Post Office Protocol

PKI Public-Key Infrastructure

POP3 Post Office Protocol v3

SMTP Simple Mail Transfer Protocol

XML Extensible Markup Language

XSD XML Schema Definition

# 5 Conventions

# None.

# 6 Overview of CMTP and CPOP commands

CMTP and CPOP have been designed as an analogy to SMTP and POP3 respectively. The purpose of these protocols is to add secure commands to the existing SMTP and POP3 protocols.

The intent of this recommendation is to overcome the limits of the email as used through SMTP and POP3. To do so, this recommendation proposes to implement two new protocols:

* Certified Mail Transfer Protocol (CMTP)
* Certified Post Office Protocol (CPOP)

The principle of these protocols is to add a set of new security commands and to update some of the existing ones within SMTP and POP3. These commands (new ones and upgrades) would enable to:

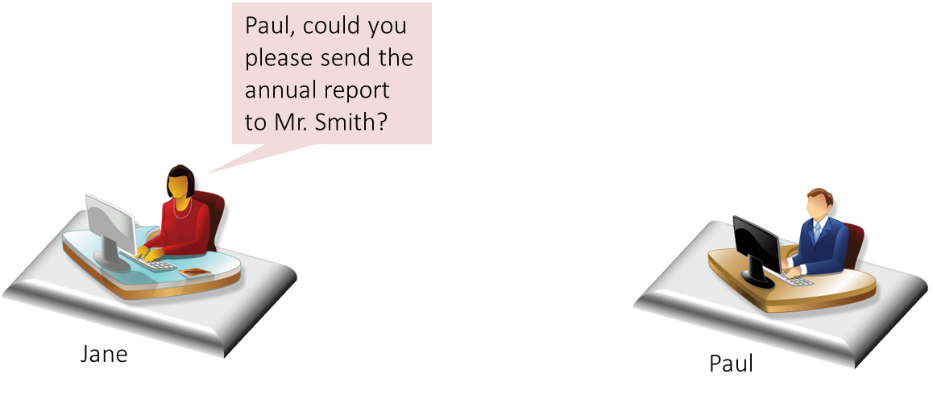
* Produce a notice of deposit electronically signed by the sender,
* Produce a notice of reception electronically signed by the recipient,
* Produce a notice of transit electronically signed by the Cmail server,
* Encrypt the message using PKI to ensure confidentiality.

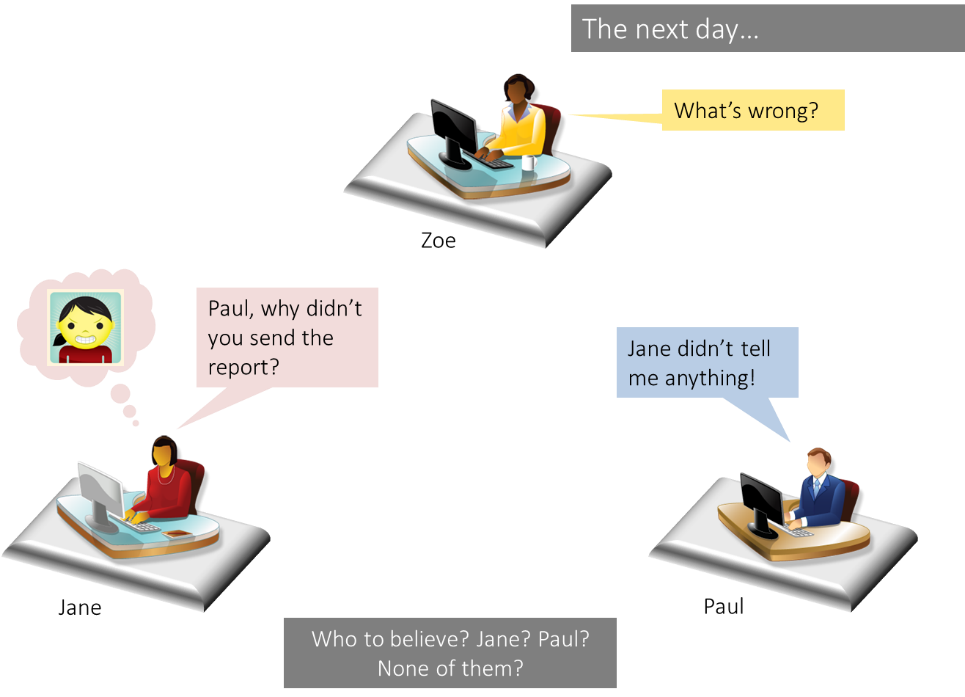
CMTP and CPOP also introduce a concept of Cmail server which role is to certify that the exchange between two parties has indeed occurred.

CMTP and CPOP are based on X509 Recommendation, electronic signature, electronic certificate and PKI.

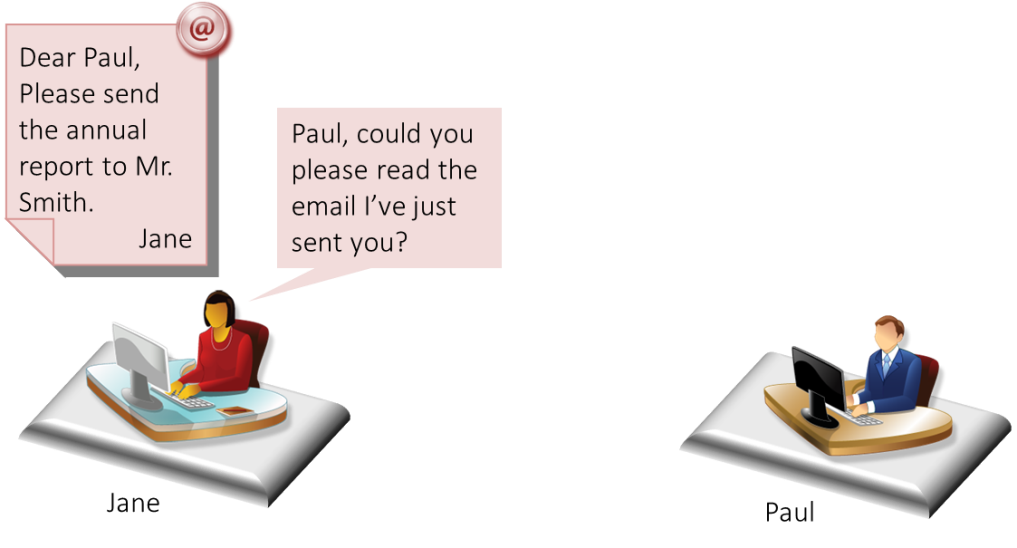
The following cartoon explains how to integrate a third party (Cmail server) to certify a bilateral exchange.

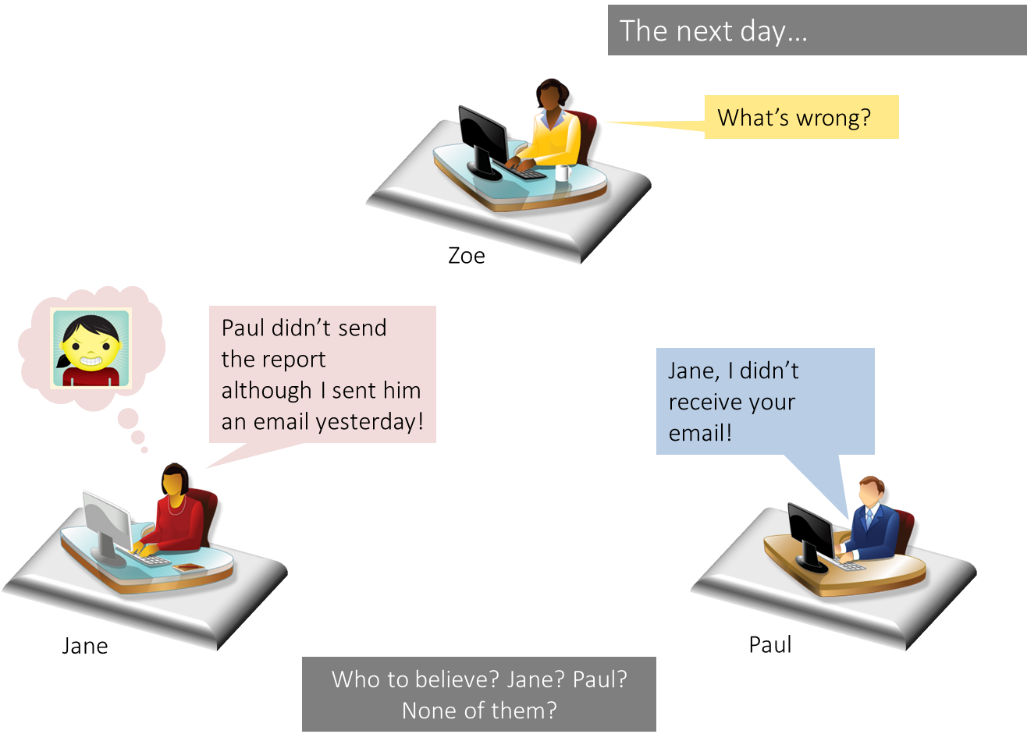
Case #1: oral communication



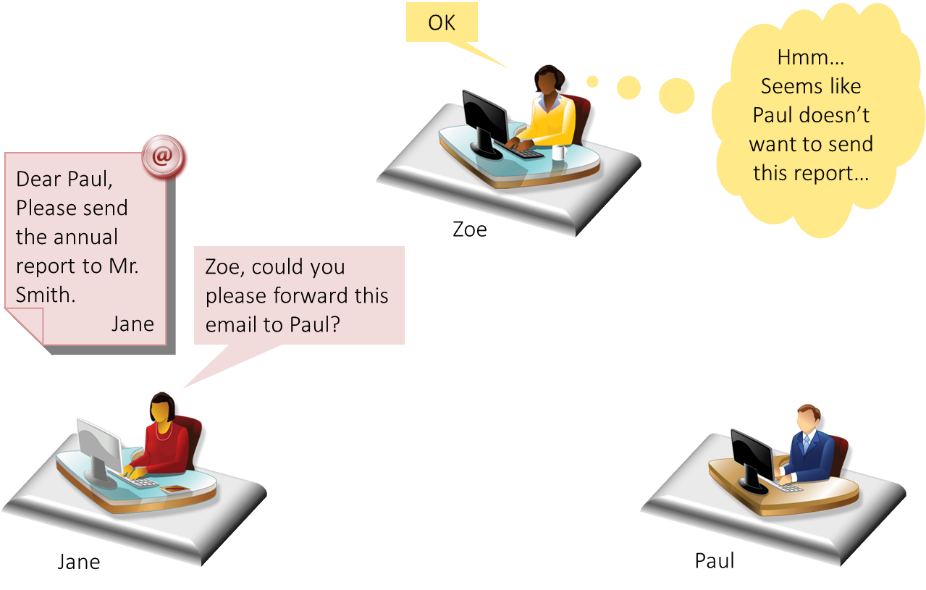


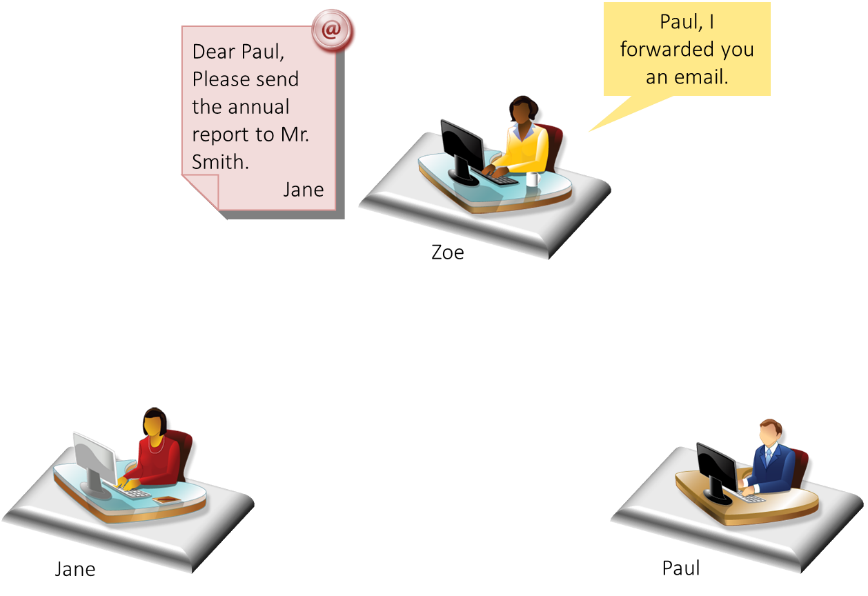
Case #2: written communication

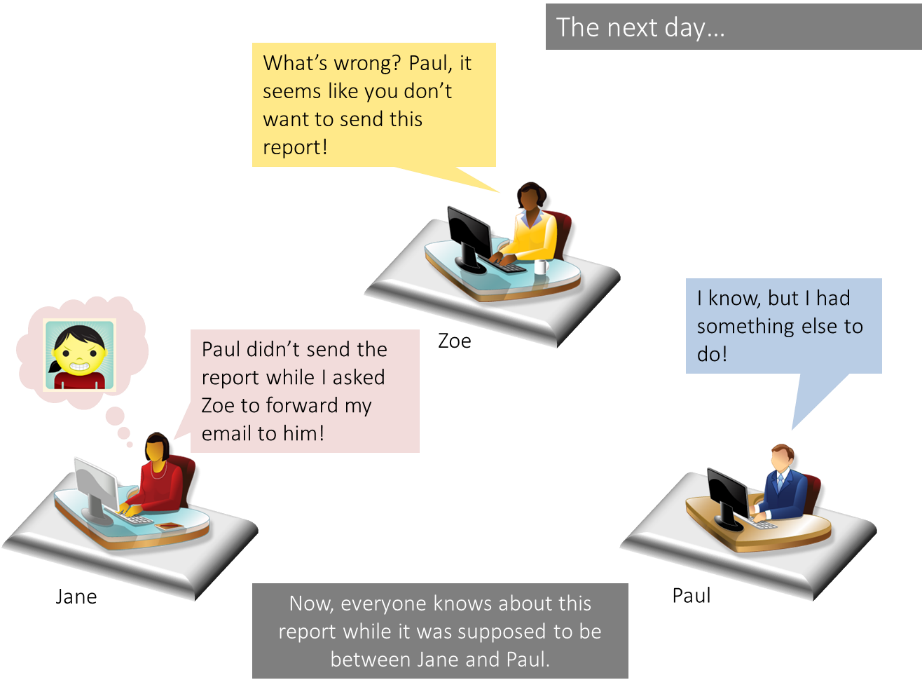




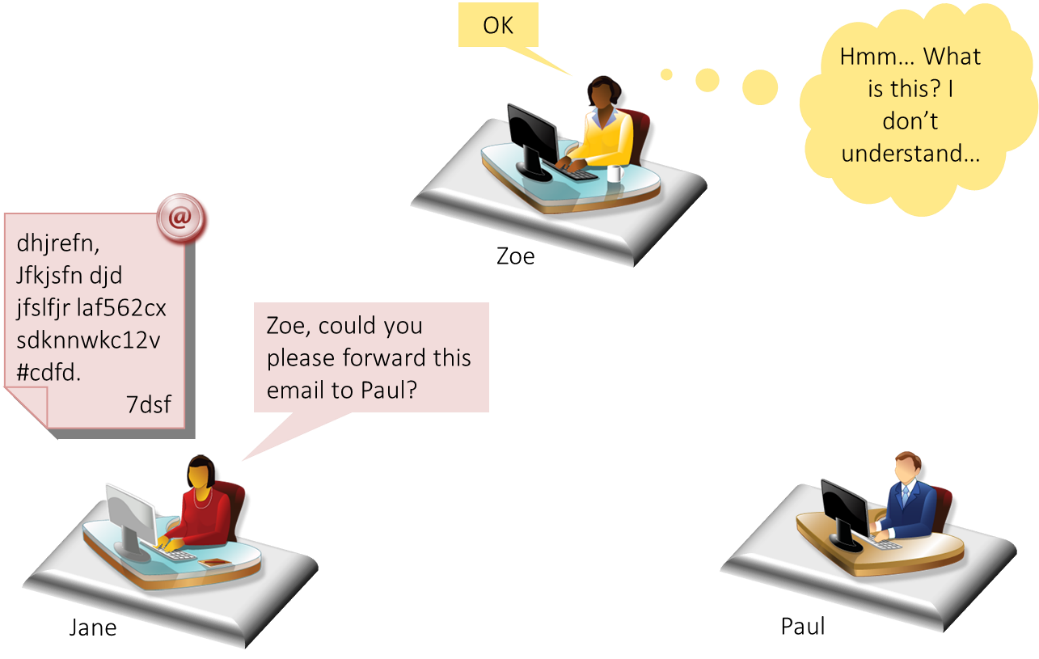
Case #3: written communication via third-party

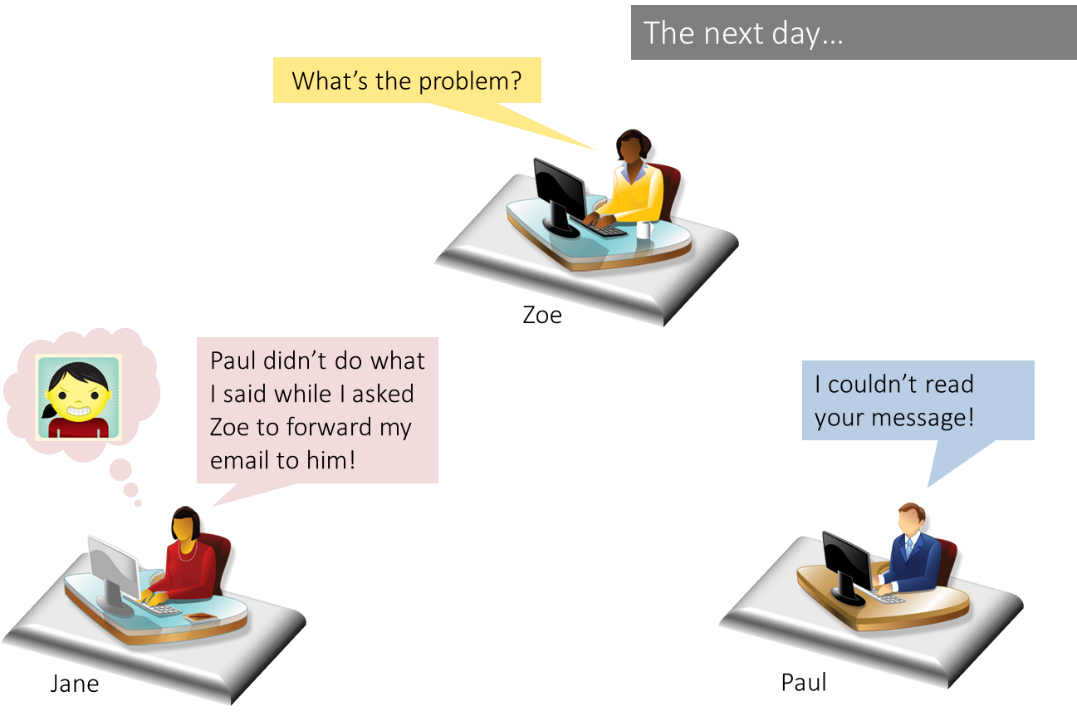




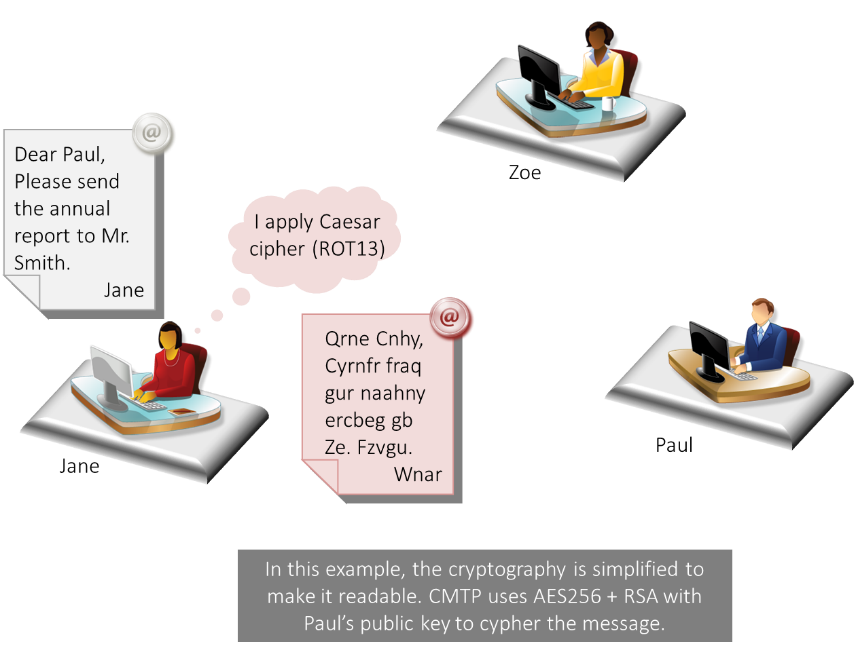


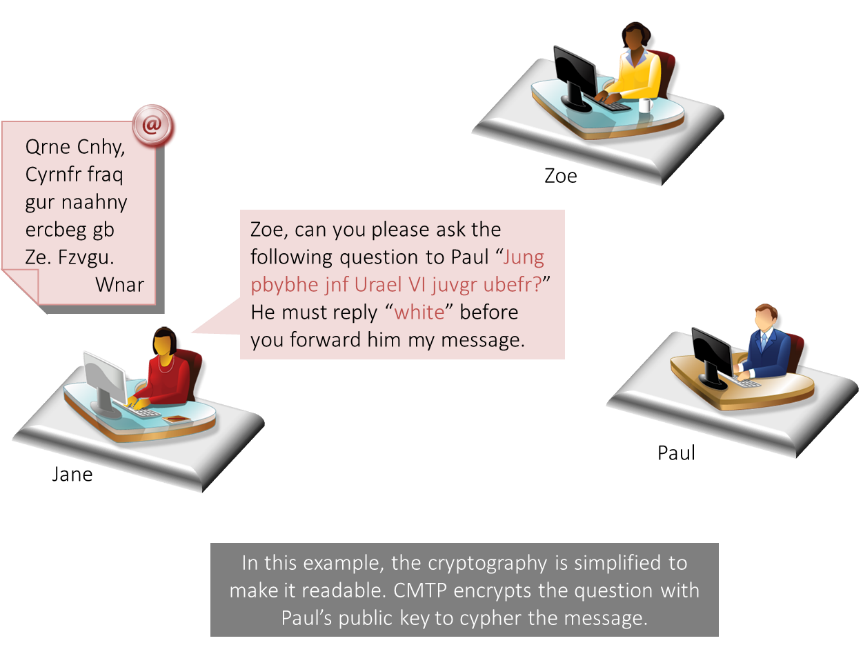
Case #4: written encrypted communication via third-party (openPGP type)

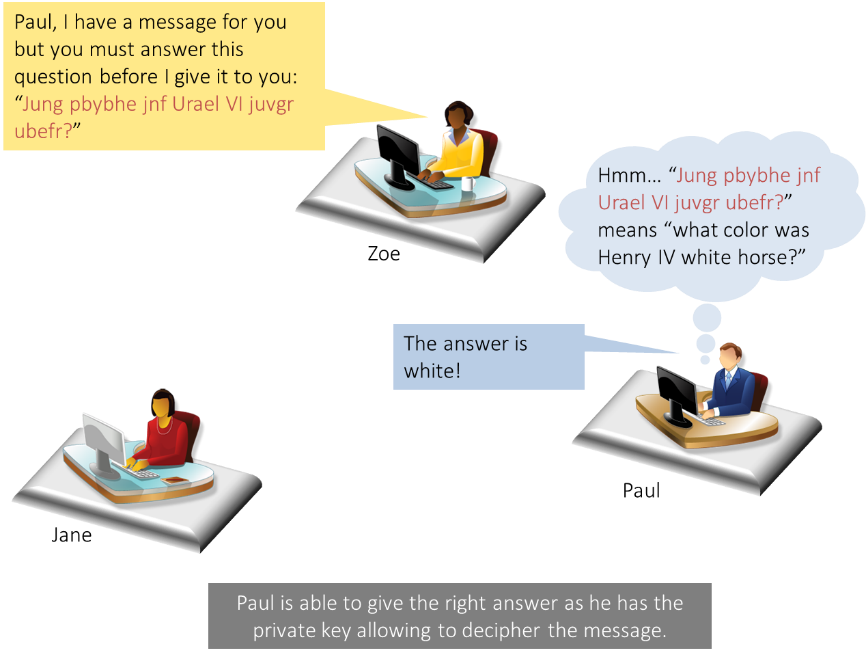


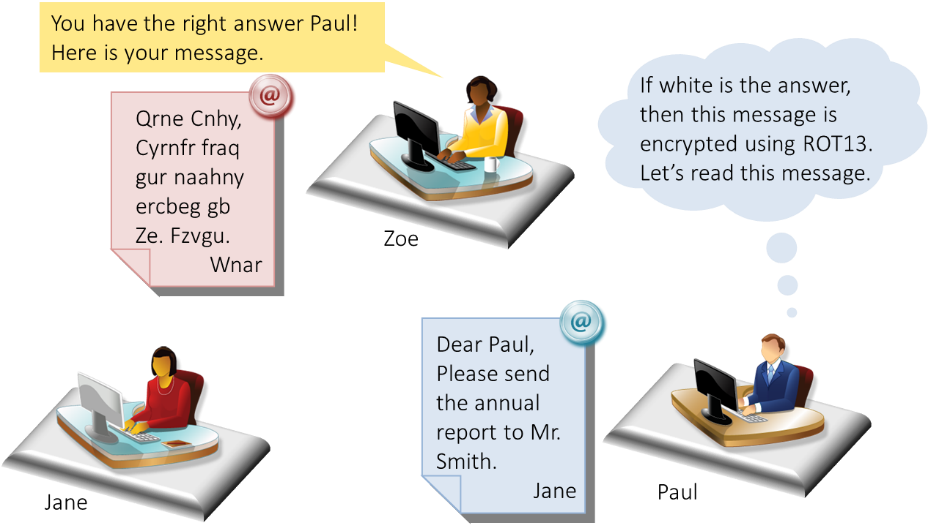


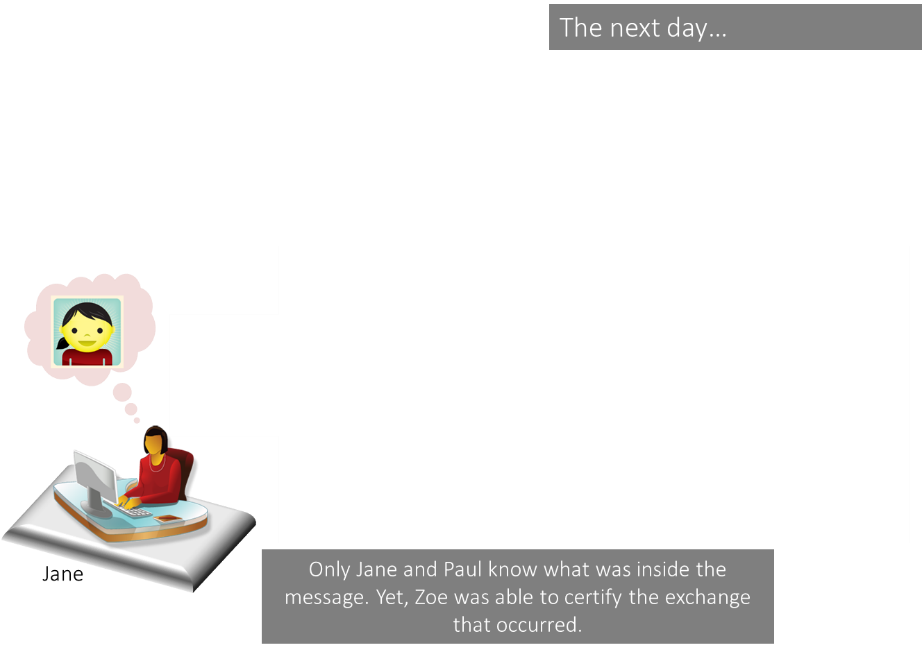
Case #5: written communication via CMTP/CPOP











Focus on CMTP/CPOP versus S/MIME:

Some work has been done in order to secure SMTP and POP3 protocols such S/MIME format, which is, with our current level of knowledge, the best way to send emails securely. However, S/MIME being a message format and not a transfer protocol, it is not possible to use it to send certified mails.

Modus operandi: SMTP + POP3 + S/MIME transaction

Sender:

* Email is written by sender,
* Email is encrypted by sender using the recipient’s public key,
* Email is signed by sender using their private key,
* S/MIME email is sent via SMTP.

Recipient:

* Recipient retrieves S/MIME email via POP3,
* Recipient checks sender’s electronic signature,
* Recipient deciphers the email and reads it.

Modus operandi: CMTP + CPOP transaction

Sender:

* Certified mail is written by sender,
* Creation of a random cypher key (hereinafter “CK”),
* Certified mail is encrypted with CK,
* Certified mail is sent to Cmail server,
* Cmail server creates a notice of deposit
* Notice of deposit is signed by Cmail server and sent back to sender,
* Sender counter-signs notice of deposit and cyphers CK with the recipient’s public key (new encrypted cypher key hereinafter “ECK”),
* Sender generates RANDOM NUMBER and computes the secret question and SECRET ANSWER (hereinafter “challenge” for the sender (e.g. challenge = SHA-1 (RANDOM NUMBER + CK) = SECRET ANSWER),
* Sender sends notice of deposit containing challenge to Cmail server.

Recipient:

* Recipient connects to the Cmail server,
* Server generates notice of reception containing the challenge and the sender’s electronic signature,
* Server sends notice of reception to recipient,
* Recipient checks sender’s electronic signature,
* Recipient deciphers ECK to CK and computes challenge,
* Recipient signs notice of reception and gives SECRET ANSWER,
* Cmail server checks challenge and sends back the cyphered certified mail,
* Recipient deciphers encrypted the certified mail using CK.

# 7 Overview of procedures



Figure 1 - Overview of protocol exchanges

Commands prefixed by “m” are used in the CMTP protocol, and commands prefixed by “c” are used in the CPOP protocol.

## 7.1 Overview of CMTP and commands

CMTP shall use the existing SMTP on table 1 commands with an addition of new specific commands on table 2.

|  |  |
| --- | --- |
| **Table 1: Standard SMTP commands** | |
| **Command** | **Command function** |
| **HELO** | Sent by a client to identify itself, usually with a domain name. |
| **EHLO** | Enables the server to identify its support for Extended Simple Mail Transfer Protocol (ESMTP) commands. |
| **MAIL FROM** | Identifies the sender of the message; used in the form “MAIL FROM:”. |
| **RCPT TO** | Identifies the message recipients; used in the form “RCPT TO:”. |
| **SIZE** | Provides a mechanism by which the SMTP server can indicate the maximum size message supported. Compliant servers must provide size extensions to indicate the maximum size message that can be accepted. Clients should not send messages that are larger than the size indicated by the server. |
| **DATA** | Sent by a client to initiate the transfer of message content. |
| **RSET** | Nullifies the entire message transaction and resets the buffer. |
| **VRFY** | Verifies that a mailbox is available for message delivery; for example, VRFY Ted verifies that a mailbox for Ted resides on the local server. |
| **HELP** | Returns a list of commands that are supported by the SMTP service. |
| **QUIT** | Terminates the session. |

|  |  |
| --- | --- |
| **Table 2: Additional and modified SMTP commands** | |
| **Command** | **Command function** |
| **CELO**  Additional | Enables the server to identify its processing of CMTP (Certified Mail Transfer Protocol) commands. |
| **DELV**  Additional | Identifies the delivery mode: certifiedMail |
| **MAIL FROM**  Modified | Identifies the sender of the message; used as “MAIL FROM:”. If the account does exist on the server, then it sends back a base64 of the public key certificate of the known sender. |
| **RCPT TO** Modified | Identifies the recipients of the message; used under “RCPT TO:” format. If the account exists on the server, then it sends back a base64 of the public key certificate of the known sender. If the account exists on another CMTP service with which key exchanges have been made, then the server questions the second server and sends a base64 of the public key certificate belonging to the known recipient with the TRANSIT CHCK RCPT command. |
| **CHCK RCPT** Additional | Sent only if the recipient is attached to another Cmail server than the Cmail server for the sender. |
| **DATA** Modified | Sent by a client to initiate the transfer of message content. The server sends back in a notice of deposit signed by the server and to be signed by the sender. |
| **DEPO** Additional | Sent by a client to initiate the transfer of the notice of deposit content signed by the server and countersigned by the sender. |
| **SEND EVLP** Additional | Forwards envelope from one Cmail server to another. |

## 7.2 Overview of CPOP and commands

CPOP shall use existing POP3 on table 3 commands with an addition of new specific commands on table 4.

|  |  |
| --- | --- |
| **Table 3: Standard POP3 commands** | |
| **Command** | **Command function** |
| **USER** | Used to specify the name of the user you wish to logon as. |
| **PASS** | This is the password of the user who is logging on |
| **STAT** | The STAT command is similar to the LIST command in that the server will return the number of messages in the Mail drop along with the total space (in octets) that those messages occupy. However, this is all the information returned by this command. To display messages, either the RETR or LIST commands must be invoked. |
| **LIST** | Used to list messages and their combined size. For example, invoking the LIST command with no parameters will return 2 +OK messages (320 octets). This means that there are two messages in the user's mail drop taking a total of 320 octets of space. |
| **RETR** | Where N is a number between 1 and the last number returned by the LIST command. This command may not be used to retrieve a message that has been marked as deleted. |
| **DELE** | This command is used to delete a message. It does not actually delete the message but simply sets a delete flag. The message is actually deleted when the client gracefully breaks the connection to the server using the QUIT command. In the case where there is a crash or immediate connection termination by the client, the messages are not deleted. |
| **RSET** | Used by the client to reset the state of messages marked for deletion. |
| **NOOP** | Null or NO OPeration. In this case, the server does nothing. |
| **HELP** | Returns a list of command that is supported by the POP3 service. |
| **QUIT** | Terminates the session. |

|  |  |
| --- | --- |
| **Table 4: Additional and modified POP3 commands** | |
| **Command** | **Command function** |
| **LIST** Modified | Used to list messages and their combined size. For example, invoking the LIST command with no parameters will return 2 +OK messages (320 octets), and the list of messages: id, length and delivery mode (if any) like CertifiedMail. |
| **RETR** Modified | Where N is a number between 1 and the last number returned by the LIST command. This command may not be used to retrieve a message that has been marked as deleted. If there is no delivery type, the server sends the email in mime encoding. If delivery mode is defined, the server processes the message specifically. For example with CertifiedMail, the server challenges the recipient before sending the envelope by using RCPT command. |
| **CHLG RESP** Additional | Sent by the client to give notice of reception for the message and give the reply to the secret question. If the reply is good then the server sends back the MIME envelope. |
| **SEND NORP** Additional | Sends the signed notice of reception. |

# 8 Detailed CMTP specification

Below is an explanation for m1 to m18 of the figure 1.

## 8.1 CELO: Ask for delivery type list

The command type is sent as a SMTP message, similar to the HELO command, followed by a fully qualified domain name. Its purpose is to retrieve a list of delivery types.

## 8.2 Delivery type list

The delivery type list is given in response to the CELO command. It is in SMTP format with the following content (case insensitive):

250-<Fully qualified domain name of the Cmail server>

250-8BITMIME

250-Delivery-Types CertifiedMail <other delivery types>

250 OK

This Recommendation only makes specification for CertifiedMail. Future editions may specify other delivery types.

## 8.3 Selected delivery type

This message identifies the delivery type of the ones specified in the delivery type list. It has the following (SMTP) format:

DELV <delivery type>

## 8.4 Delivery type acknowledgement

In the case the selected delivery type is accepted, this message has the following SMTP format (case insensitive):

250 Delivery-Type <delivery type>OK

The following response is given in case of a syntax error in selected delivery message

501 Syntax: DELV <delivery type>

The following response is given when the selected delivery message was issued out of sequence:

501 Syntax: use CELO command first

The following response is given when the selected delivery message was unknown:

501 Unknown Delivery-Type: <delivery type>

## 8.5 Sender's email address

This message is sent to the Cmail server to request sending a certified email and optionally to request the sender's public-key certificate from the Cmail server.

MAIL FROM: <sender's email address> [CertificateRequested]

## 8.6 Sender's email acknowledgement

This message is sent to confirm that the sender's email address exists in the Cmail server database. If the sender requested its public-key certificate, the sender's public-key certificate is included:

[250 User-Certificate: <public-key certificate encoded in Base64>]

250 OK

## 8.7 Ask for sending email to recipient

This message is sent to the Cmail server to request sending one certified email to the recipient and optionally to request the recipient's public-key certificate from the Cmail server.

RCPT TO: <recipient's email address> [CertificateRequested]

This command may be used as many times as necessary in order to add each recipient if several. The information indicating whether the recipient is “To” or “Cc” is contained in the header of the envelope (RFC 821, chapter 3.6 relaying). “Cci” recipients are not allowed.

## 8.8 Check recipient's email address by the remote Cmail server

This message is only sent if the recipient is attached to another Cmail server than the Cmail server for the sender. It is sent from the sender's Cmail server to the recipient's Cmail server to check the validity of the email address and optionally to request the recipient's public-key certificate.

CHCK RCPT: <recipient's email address> [CertificateRequested]

## 8.9 Recipient's email address acknowledgement

This message is sent in response to "Check recipient's email address by the remote Cmail server".

The following confirms the email address and includes the recipient's public-key certificate if so requested:

[250 User-Certificate: <public-key certificate encoded in Base64>]

250 OK

In case the email address cannot be confirmed, the following error messages may be sent:

503 Sender already specified

shall be sent if it is a response to a duplicate request.

501 Syntax: CHCK RCPT: <address>

shall be sent if there is a syntax error in the recipient's email address

501 Syntax: CHCK RCPT: <address> Error in parameters <parameter>

shall be sent if the parameter after the email address was not recognised.

553 <email address> Invalid email address.

shall be sent if the email address does not exist at the remote Cmail server.

## 8.10 Recipient's email acknowledgement

This message is sent to confirm that the recipient's email address exists. If the sender requested the recipient's public-key certificate, the recipient’s public-key certificate is included:

The following confirms the email address and includes the recipient's public-key certificate if so requested:

[250 User-Certificate: <public-key certificate encoded in Base64>]

250 OK

In case the email address cannot be confirmed, the following error messages may be sent:

503 Error: need MAIL FROM command

shall be sent if the message was sent out of sequence.

452 Error: too many recipients

shall be sent if too many recipients were specified.

501-6.1.1 Syntax: RCPT TO: <address>

shall be sent if there is a syntax error in the recipient's email address

501-6.1.2 Syntax: RCPT TO: <address> Error in parameters: <parameters>

shall be sent if the parameter after the email address was not recognised.

550-5.1.1 <email address> Invalid email address.

shall be sent if the email address does not exist.

## 8.11 Ask for sending ENVELOPE

The following format is used by the sender to ask the Cmail server permission to send data DATA

## 8.12 Ready to receive ENVELOPE

The following message is sent if the Cmail server is ready to receive data:

354 Start mail input; end with <CRLF>.<CRLF>

The following message is sent when the MAIL FROM command has not been sent

503 Error: need MAIL FROM command

The following message is sent when the RCPT TO command has not been sent:

503 Error: need RCPT TO command

The following message is sent when the DELV command has not been sent

503 Error: need DELV command

## 8.13 ENVELOPE

The client shall:

1. generate a random symmetric cipher key (called RSCK), e.g. AES 256;

2. encrypt the body of the message and attachments, if any, using this key;

3. build a MIME message containing a part named ENVELOPE which contains the encrypted message (see MIME IETF RFC 2045);

4. end the message with <CR><LF>.<CR><LF>; and

5. send the MIME message.

## 8.14 Server signed notice of deposit

250 Notice-of-deposit:

<notice of deposit signed by the Cmail server encoded in base64 >

250 Ok

Server generates a notice of deposit containing information about the envelope (envelope id, delivery type and mime hash), and signs it with its private key.

## 8.15 Sender and server signed notice of deposit

The sender shall:

1. decode the received notice of deposit;

2. build challenge for each recipients;

3. sign the server signed notice of deposit using its own private key;

4. encode the result in base64; and

5. transmit it to the Cmail server using:

DEPO <notice of deposit base64 encoded>

The challenge is defined in A.2.

The challenge contains the **SecretQuestion**, **CipherEnvelopeKey**, and the public key certificate of the recipient.

**SecretQuestion**: is composed by a **Request** and a **Response**.

The Request may contains a **RandomNumber**. The Response contains the **AlgorithmIdentifier** to recalculate by the sender in order to receive the ENVELOPE. This **AlgorithmIdentifier** identify the algorithm used to compute hash. The challenge consists to, first recover the cipher key RSCK, ciphered by the public key of the recipient. Then concatenate the **RandomNumber** and the RSCK, and compute hash to build the response.

Example of a challenge in XML:

<Entity EmailAddress="john.doe@example.org" Type="to">

<SecretQuestion>

<Request RandomNumber="30987497498789739837"/>

<Response AlgorithmIdentifier="2.16.840.1.101.3.4.2.1" Encoding="base64">5mYZWhtl0yxBa/wl7VLiiQ=</response>

</SecretQuestion>

<CipherEnvelopeKey Algorithm="AES" CipheredKey="RSA" Encoding="base64-DER" KeySize="256">UjBg…b1PHDOOM4IFnTpzHn9TQ==</cipherEnvelopeKey>

<Certificate Encoding="base64">MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AM…sdjn7VDBlb+WS10j2rJcAHHsUyr…

/gy7</Certificate>

</Entity>

Note 1: this challenge could use ASN.1 format DER encoding.

Note 2: the server is not able to recalculate the hash since it does not know the encryption key. However, it is the only one to know the expected result from the hash calculation.

Note 3: during the challenge with the recipient, the server sends only the secret question and waits for the recipient’s reply.

## 8.16 ENVELOPE between Cmail servers

The message defined in 8.13 is forwarded to another Cmail server only if the sender and the recipient are attached to different Cmail servers (see item m16 of figure 1).

SEND EVLP <MIME message>

## 8.17 Signed notice of transit between Cmail servers

The following format shall be used:

250 Notice-of-transit:

<notice of transit base64 encoded>

The following message is sent if the Cmail server receives Notice-of-transit

250 Ok

The following message is sent when the Notice-of-transit is incorrect:

503 Error: incorrect Notice-of-transit

Notice of transit is built by the Cmail who received the ENVELOPE.

This Cmail server generates a notice of deposit containing information about the envelope (envelope id, delivery type and mime hash), and signs it with its private key. This notice is the same as notice of deposit.

## 8.18 Signed notice of transit

The Cmail sender server shall:

1. decode the received notice of transit;

2 sign the server signed notice of transit using its own private key;

3. encode the result in base64; and

4. transmit it to the Cmail server using:

250 Signed-notice-of-transit:

<signed notice of transit base64 encoded>

250 Signed-notice-of-deposit:

<signed notice of deposit base64 encoded>

250 Ok

# 9 The Certified Post Office Protocol (CPOP)

Below is an explanation for p1 to p6 of the figure 1.

## 9.1 Ask for pending messages

Information on pending messages is performed using the procedure specified in section 5 under LIST command in IETF RFC 1939 with addition that for each line detailing a pending message, a new parameter is added indicating the delivery type if it not a standard email (see item p1 of figure 1)Example:

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200 CertifiedMail

S: .

This procedure also includes retrieving all standard emails leaving only messages tagged with delivery type on the Cmail server.

## 9.2 Challenge recipient and server signed notice of reception

For messages tagged with delivery type, the POP3 RETR command does not retrieve the message but retrieves the challenge and the server signed notice of reception base64 encoded. The client verify the digital signature and the sender certificate contained in the notice of reception.

Example:

C: RETR 2

The following message is sent if the Cmail server sends the notice of reception

S: +OK 200 octets

S: <the Cmail server sends the notice of reception including the challenge>

S: .

The following message is sent when the server cannot send the notice of reception:

503 Error: impossible to send Notice-of-reception

The Cmail server finds in the Notice of deposit, the node **Entity** related to the recipient. Then copy this node in the notice of reception and remove the content of the **Response** node included in **Entity** node.

Example, node in the notice of deposit:

<Entity EmailAddress="john.doe@example.org" Type="to">

<SecretQuestion>

<Request RandomNumber="30987497498789739837"/>

<Response AlgorithmIdentifier="2.16.840.1.101.3.4.2.1" Encoding="base64">5mYZWhtl0yxBa/wl7VLiiQ=</response>

</SecretQuestion>

<CipherEnvelopeKey Algorithm="AES" CipheredKey="RSA" Encoding="base64-DER" KeySize="256">UjBg…b1PHDOOM4IFnTpzHn9TQ==</cipherEnvelopeKey>

<Certificate Encoding="base64">MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AM…sdjn7VDBlb+WS10j2rJcAHHsUyr…

/gy7</Certificate>

</Entity>

Note: this challenge could use ASN.1 format DER encoding

And same node copied in the notice of reception:

<Entity EmailAddress="john.doe@example.org" Type="to">

<SecretQuestion>

<Request RandomNumber="30987497498789739837"/>

<Response AlgorithmIdentifier="2.16.840.1.101.3.4.2.1" Encoding="base64" />

</SecretQuestion>

<CipherEnvelopeKey Algorithm="AES" CipheredKey="RSA" Encoding="base64-DER" KeySize="256">UjBg…b1PHDOOM4IFnTpzHn9TQ==</cipherEnvelopeKey>

<Certificate Encoding="base64">MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AM…sdjn7VDBlb+WS10j2rJcAHHsUyr…

/gy7</Certificate>

</Entity>

Note: this challenge could use ASN.1 format DER encoding

## 9.3 Challenge response and recipient and server signed notice of reception

The recipient shall:

1. decode the received notice of reception;

2. retrieve the RSCK;

3. compute challenge response,

4. sign the server signed notice of reception using its own private key;

5. encode the result in base64; and

6. transmit it to the Cmail server using:

CHLG RESP <challenge response and recipient and server signed notice of reception>

The recipient deciphers the message as follow:

<Entity EmailAddress="john.doe@example.org" Type="to">

<SecretQuestion>

<Request RandomNumber="30987497498789739837"/>

<Response AlgorithmIdentifier="2.16.840.1.101.3.4.2.1" Encoding="base64"></response>

</SecretQuestion>

<CipherEnvelopeKey Algorithm="AES" CipheredKey="RSA" Encoding="base64-DER" KeySize="256">UjBg…b1PHDOOM4IFnTpzHn9TQ==</cipherEnvelopeKey>

<Certificate Encoding="base64">MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AM…sdjn7VDBlb+WS10j2rJcAHHsUyr…

/gy7</Certificate>

</Entity>

Note: this challenge could use ASN.1 format DER encoding

Recipient recover RSCK using his private key by decipher the content of node **CipherEnvelopeKey**. Then he concatenate **RandomNumber** and RSCK, hash it using the **AlgorithmIdentifier** defined, and get the result of the **SecretQuestion**.

Recipient copy this result in the signed notice of reception, sign it and send it to the Cmail server.

## 9.4 ENVELOPE

If the challenge is OK, the Cmail server sends the ENVELOPE like the result of the POP3 command RETR. The recipient now has the message ant the key to open it.

Editor's comment: Error situations to be included

The following message is sent when the server cannot send the ENVELOPE:

503 Error: impossible to send ENVELOPE

## 9.5 Recipient and server signed notice of reception between Cmail servers (optional)

This message is only sent if the sender and the recipient are attached to different Cmail servers.

SEND NORP <base64 encoded Recipient and server signed notice of reception>

## 9.6 Recipient and server signed notice of reception (optional)

This message is only sent if the sender and the recipient are attached to different Cmail servers.

SEND NORP <base64 encoded Recipient and server signed notice of reception>

# 10 Envelope and Notices format

## 10.1 Notice of deposit

A notice of deposit contains information about sender, envelope, and is co-signed by Cmail server and the sender.

It is an evidence of deposit for the sender who can use it in case of litigation.

The formal specification of the notice of deposit can be found in annex A.

Example: file “1373360283931.deposit.notice”

|  |
| --- |
| Received: from localhost ([127.0.0.1])  by begmeil  with SMTP (SubEthaSMTP null) id HIWV8HF9  for laura.prin@legalbox.com;  Tue, 09 Jul 2013 10:58:14 +0200 (CEST)  Content-Type: application/octet-stream  Content-Transfer-Encoding: base64  Content-Disposition: attachment; filename=depositNotice.xml  PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9Im5vIj8+Cjxs  ZXR0ZXJEZXBvc2l0UG9zdG1hcms+CiAgPG9wZXJhdG9yUG9zdG1hcms+CiAgICA8ZW52ZWxvcElk  bG9wZWQtc2lnbmF0dXJlIi8+CiAgICAgICAgICA8L1RyYW5zZm9ybXM+CiAgICAgICAgICA8RGln  ...  ICAgPFJTQUtleVZhbHVlPgogICAgICAgICAgICA8TW9kdWx1cz5tMkFSUURXUGJBMmgvMzJEQWs4  ICAgICAgICAgIDxFeHBvbmVudD5BUUFCPC9FeHBvbmVudD4KICAgICAgICAgIDwvUlNBS2V5VmFs  dWU+CiAgICAgICAgPC9LZXlWYWx1ZT4KICAgICAgPC9LZXlJbmZvPgogICAgPC9TaWduYXR1cmU+  CiAgPC9lbnZlbG9wSW5mb3JtYXRpb24+CjwvbGV0dGVyRGVwb3NpdFBvc3RtYXJrPgo= |

## 10.2 Notice of reception

Notice of reception contains information about sender, envelope, the challenge to open the envelope, and is co-signed by Cmail server and the recipient.

It is an evidence of reception for the sender who can use it in case of litigation.

The formal specification of the notice of reception can be found in annex A.

Example: file “[1373360283931.laura.prin@legalbox.com.receipt.notice](mailto:1373360283931.laura.prin@legalbox.com.receipt.notice)”

|  |
| --- |
| Received: from TODO get hostname ([127.0.0.1])  by localhost  with SMTP (LegalBox POP Server v1.0) id HIWX27L5  for laura.prin@legalbox.com;  Tue, 09 Jul 2013 11:49:01 +0200 (CEST)  Content-Type: application/octet-stream  Content-Transfer-Encoding: base64  Content-Disposition: attachment; filename=receiptNotice.xml  PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9Im5vIj8+Cjxs  ZXR0ZXJEZXBvc2l0UG9zdG1hcms+CiAgPG9wZXJhdG9yUG9zdG1hcms+CiAgICA8ZW52ZWxvcElk  PjEzNzMzNjAyODM5MzE8L2VudmVsb3BJZD4KICAgIDxkZWxpdmVyeU1vZGU+Y2VydGlmaWVkTGV0  dGVyPC9kZWxpdmVyeU1vZGU+CiAgICA8bWltZU1lc3NhZ2VIYXNoPgogICAgICA8c2hhMT5hNTVk  ZDhmYWU0Mzg2M2VmYWRmMWY3ZjM3MmEwYmU1MmEwMGRhYTFkPC9zaGExPgogICAgPC9taW1lTWVz  ...  MDkwSDl0NFVkTTdWVU92bjY3WlU2aTJvVSt3b3lGR2tYMDJ3YkVMM2pDYmpJCm5VR1BwUGpoT3Zo  dzNPTy9mYmhKVk13dkM2NXB1MTl1cnA2M05kS0tHNlBuNjZtQkVnUldxZ2cvTVBITmZmWkhrOXFs  WExSSXhETi8Kb0ZnS285RmI0NExlSzBnZ3Vyb1Y2azNicm1TeGM1UnpYVWNxTzdwbldUN0FoNFl6  WXJJUHdYL1hjS1VqbXYxZi9JZjQ5VHVnWGtLcgpodklyOG9qUkdQcEdPdlB4cWR5QWNQR1BOUVRY  NFJrc29kSEVwdz09PC9Nb2R1bHVzPgogICAgICAgICAgICA8RXhwb25lbnQ+QVFBQjwvRXhwb25l  bnQ+CiAgICAgICAgICA8L1JTQUtleVZhbHVlPgogICAgICAgIDwvS2V5VmFsdWU+CiAgICAgIDwv  S2V5SW5mbz4KICAgIDwvU2lnbmF0dXJlPgogIDwvcmVjaXBpZW50Q2hhbGVuZ2U+CjwvbGV0dGVy  RGVwb3NpdFBvc3RtYXJrPgo= |

## 10.3 Notice of transit

Notice of deposit contains information about sender, envelope, the challenge to open the envelope, and is co-signed by Cmail servers.

It is an evidence of transit for the sender who can use it in case of litigation.

The formal specification of the notice of transit can be found in annex A.

Example: file “[1373360283931.laura.prin@legalbox.com.receipt.notice](mailto:1373360283931.laura.prin@legalbox.com.receipt.notice)”

|  |
| --- |
| Received: from TODO get hostname ([127.0.0.1])  by localhost  with SMTP (LegalBox POP Server v1.0) id HIWX27L5  for laura.prin@legalbox.com;  Tue, 09 Jul 2013 11:49:01 +0200 (CEST)  Content-Type: application/octet-stream  Content-Transfer-Encoding: base64  Content-Disposition: attachment; filename=receiptNotice.xml  PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9Im5vIj8+Cjxs  ZXR0ZXJEZXBvc2l0UG9zdG1hcms+CiAgPG9wZXJhdG9yUG9zdG1hcms+CiAgICA8ZW52ZWxvcElk  PjEzNzMzNjAyODM5MzE8L2VudmVsb3BJZD4KICAgIDxkZWxpdmVyeU1vZGU+Y2VydGlmaWVkTGV0  dGVyPC9kZWxpdmVyeU1vZGU+CiAgICA8bWltZU1lc3NhZ2VIYXNoPgogICAgICA8c2hhMT5hNTVk  ZDhmYWU0Mzg2M2VmYWRmMWY3ZjM3MmEwYmU1MmEwMGRhYTFkPC9zaGExPgogICAgPC9taW1lTWVz  ...  MDkwSDl0NFVkTTdWVU92bjY3WlU2aTJvVSt3b3lGR2tYMDJ3YkVMM2pDYmpJCm5VR1BwUGpoT3Zo  dzNPTy9mYmhKVk13dkM2NXB1MTl1cnA2M05kS0tHNlBuNjZtQkVnUldxZ2cvTVBITmZmWkhrOXFs  WExSSXhETi8Kb0ZnS285RmI0NExlSzBnZ3Vyb1Y2azNicm1TeGM1UnpYVWNxTzdwbldUN0FoNFl6  WXJJUHdYL1hjS1VqbXYxZi9JZjQ5VHVnWGtLcgpodklyOG9qUkdQcEdPdlB4cWR5QWNQR1BOUVRY  NFJrc29kSEVwdz09PC9Nb2R1bHVzPgogICAgICAgICAgICA8RXhwb25lbnQ+QVFBQjwvRXhwb25l  bnQ+CiAgICAgICAgICA8L1JTQUtleVZhbHVlPgogICAgICAgIDwvS2V5VmFsdWU+CiAgICAgIDwv  S2V5SW5mbz4KICAgIDwvU2lnbmF0dXJlPgogIDwvcmVjaXBpZW50Q2hhbGVuZ2U+CjwvbGV0dGVy  RGVwb3NpdFBvc3RtYXJrPgo= |

## 10.4 ENVELOPE

ENVELOPE a MIME Message containing the mail content ciphered by AES encryption

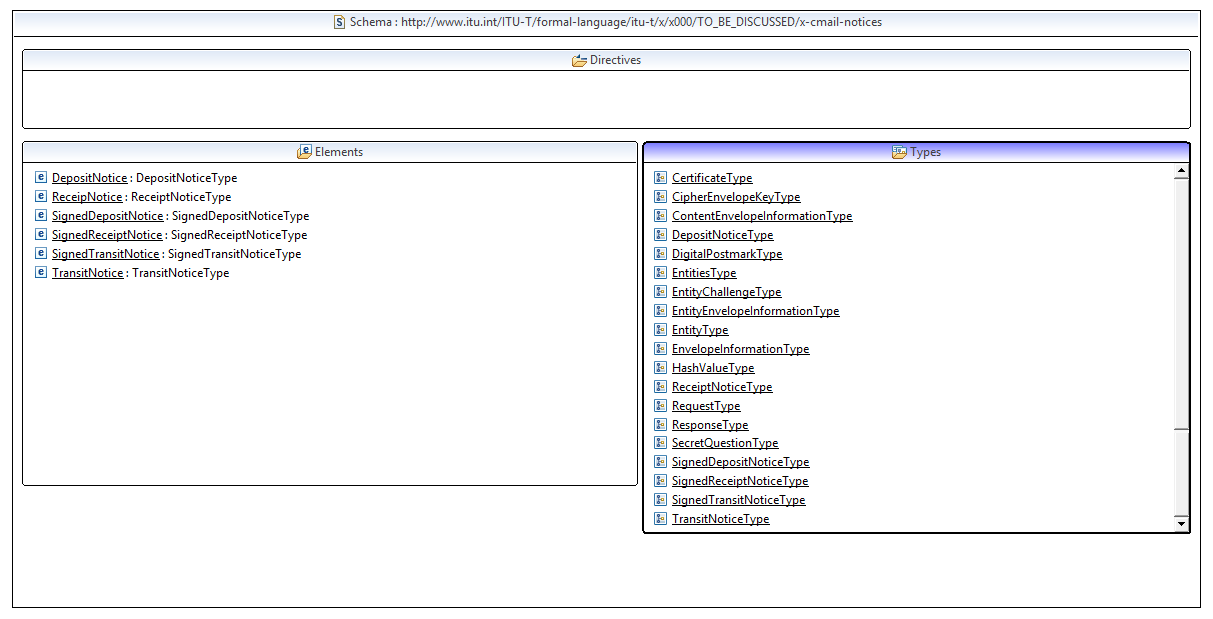
Example: file “1373360283931.certifiedLetter.msg”

|  |
| --- |
| Received: from localhost ([127.0.0.1])  by begmeil  with SMTP (SubEthaSMTP null) id HIWV8HF9  for laura.prin@legalbox.com;  Tue, 09 Jul 2013 10:58:03 +0200 (CEST)  Date: Tue, 9 Jul 2013 10:57:51 +0200 (CEST)  From: david.keller@legalbox.com  To: laura.prin@legalbox.com  Message-ID: proto\_cmtp\_1373360269856  Subject: =?UTF-8?Q?Bienvenue\_=C3=A0\_CMTP!?=  MIME-Version: 1.0  Content-Type: multipart/mixed;  boundary="----=\_Part\_1\_1013939722.1373360271613"  ------=\_Part\_1\_1013939722.1373360271613  Content-Type: multipart/mixed;  boundary="----=\_Part\_0\_2062834323.1373360271584"  ------=\_Part\_0\_2062834323.1373360271584  Content-Type: application/octet-stream  Content-Transfer-Encoding: base64  Content-Disposition: attachment; filename=envelop  RG44gUlyr1A/L+ps0R+yKMUpgPcJACmcRQdLZSMoLnm07gtRataSAWkG5qnc/f5Q  ------=\_Part\_0\_2062834323.1373360271584--  ------=\_Part\_1\_1013939722.1373360271613-- |

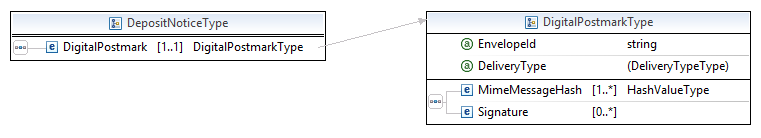
Annex A  
  
Notices in XSD

(This annex forms an integral part of this Recommendation.)

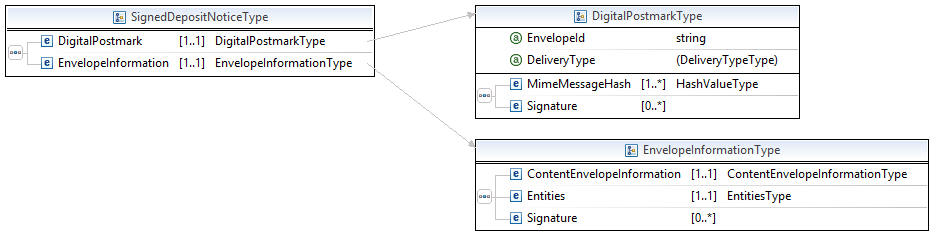
## A.1 XSD overview



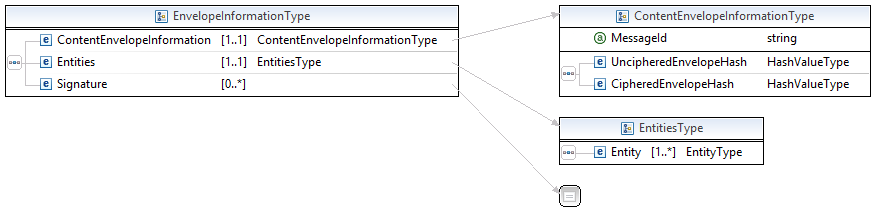
Notice of deposit



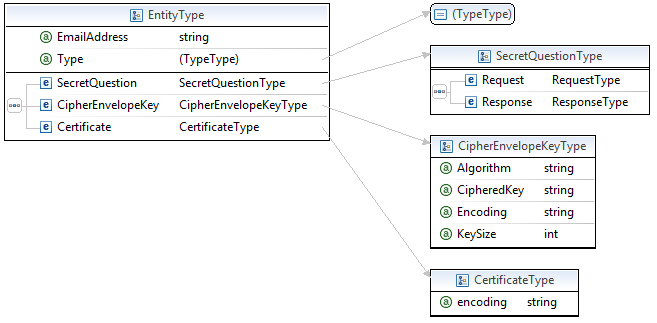
Signed notice of deposit



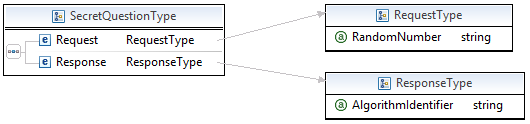
Envelope Information Type



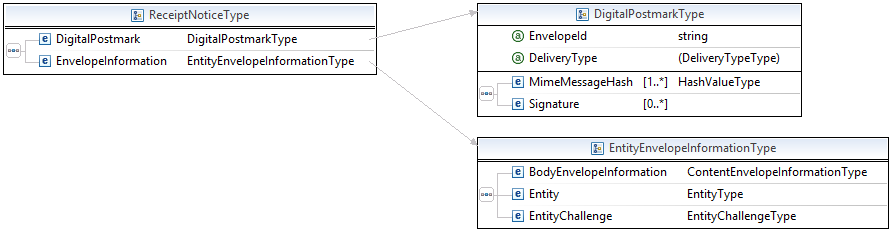
Entity Type



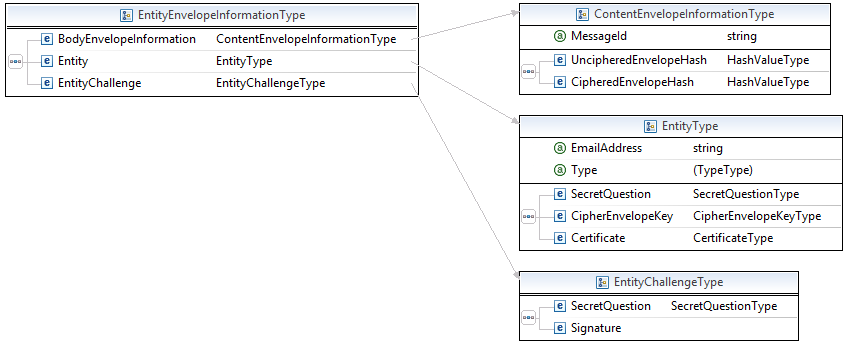
Challenge



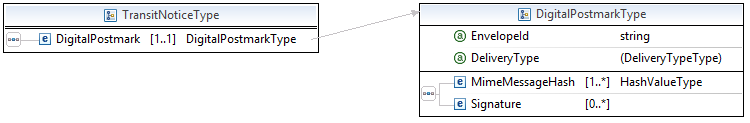
Notice of reception



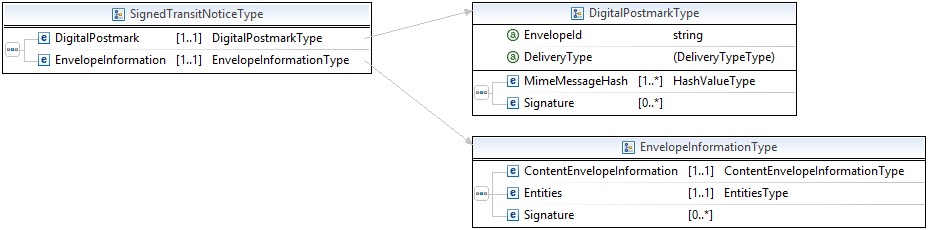
Recipient’s answer to challenge



Notice of transit



Signed notice of transit



## A.2 Formal protocol specification in XSD

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<schema targetNamespace=*"http://www.itu.int/ITU-T/formal-language/TO\_BE\_DISCUSSED/x-cmail-notices"*

elementFormDefault=*"qualified"* xmlns=*"http://www.w3.org/2001/XMLSchema"* xmlns:tns=*"http://www.itu.int/ITU-T/formal-language/TO\_BE\_DISCUSSED/x-cmail-notices"* xmlns:ds=*"http://www.w3.org/2000/09/xmldsig#"*>

<import namespace=*"http://www.w3.org/2009/xmldsig11#"*

schemaLocation=*"http://www.w3.org/TR/xmldsig-core1/xmldsig11-schema.xsd"* />

<import namespace=*"http://www.w3.org/2009/xmldsig-properties"*

schemaLocation=*"http://www.w3.org/TR/xmldsig-properties/xmldsig-properties.xsd"* />

<import schemaLocation=*"http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"* namespace=*"http://www.w3.org/2000/09/xmldsig#"*></import>

<element name=*"DepositNotice"* type=*"tns:DepositNoticeType"*></element>

<element name=*"SignedDepositNotice"* type=*"tns:SignedDepositNoticeType"*></element>

<element name=*"TransitNotice"* type=*"tns:TransitNoticeType"*></element>

<element name=*"SignedTransitNotice"* type=*"tns:SignedTransitNoticeType"*></element>

<element name=*"ReceipNotice"* type=*"tns:ReceiptNoticeType"*></element>

<element name=*"SignedReceiptNotice"* type=*"tns:SignedReceiptNoticeType"*></element>

<complexType name=*"DigitalPostmarkType"*>

<sequence>

<element name=*"MimeMessageHash"* type=*"tns:HashValueType"*

maxOccurs=*"unbounded"* minOccurs=*"1"*>

</element>

<element name=*"Signature"* type=*"ds:SignatureType"*

maxOccurs=*"unbounded"* minOccurs=*"0"*>

</element>

</sequence>

<attribute name=*"EnvelopeId"* type=*"string"* use=*"required"*></attribute>

<attribute name=*"DeliveryType"* use=*"required"*>

<simpleType>

<restriction base=*"string"*>

<enumeration value=*"CertifiedMail"*></enumeration>

</restriction>

</simpleType>

</attribute>

</complexType>

<complexType name=*"EnvelopeInformationType"*>

<sequence>

<element name=*"ContentEnvelopeInformation"*

type=*"tns:ContentEnvelopeInformationType"* maxOccurs=*"1"* minOccurs=*"1"*>

</element>

<element name=*"Entities"* type=*"tns:EntitiesType"*

maxOccurs=*"1"* minOccurs=*"1"*>

</element>

<element name=*"Signature"* type=*"ds:SignatureType"*

maxOccurs=*"unbounded"* minOccurs=*"0"*>

</element>

</sequence>

</complexType>

<complexType name=*"ContentEnvelopeInformationType"*>

<sequence>

<element name=*"UncipheredEnvelopeHash"* type=*"tns:HashValueType"*></element>

<element name=*"CipheredEnvelopeHash"* type=*"tns:HashValueType"*></element>

</sequence>

<attribute name=*"MessageId"* type=*"string"*></attribute>

</complexType>

<complexType name=*"SecretQuestionType"*>

<sequence>

<element name=*"Request"* type=*"tns:RequestType"*></element>

<element name=*"Response"* type=*"tns:ResponseType"*></element>

</sequence>

</complexType>

<complexType name=*"EntityType"*>

<sequence>

<element name=*"SecretQuestion"* type=*"tns:SecretQuestionType"*></element>

<element name=*"CipherEnvelopeKey"*

type=*"tns:CipherEnvelopeKeyType"*>

</element>

<element name=*"Certificate"* type=*"tns:CertificateType"*></element>

</sequence>

<attribute name=*"EmailAddress"* type=*"string"* use=*"required"*>

<annotation>

<documentation>Email address have to be in RFC 2822 format</documentation>

</annotation></attribute>

<attribute name=*"Type"* use=*"required"*>

<simpleType>

<restriction base=*"string"*>

<enumeration value=*"from"*></enumeration>

<enumeration value=*"to"*></enumeration>

<enumeration value=*"cc"*></enumeration>

<enumeration value=*"transit"*></enumeration>

</restriction>

</simpleType>

</attribute>

</complexType>

<complexType name=*"CipherEnvelopeKeyType"*>

<attribute name=*"Algorithm"* type=*"string"*></attribute>

<attribute name=*"CipheredKey"* type=*"string"*></attribute>

<attribute name=*"Encoding"* type=*"string"*></attribute>

<attribute name=*"KeySize"* type=*"int"*></attribute>

</complexType>

<complexType name=*"CertificateType"*>

<attribute name=*"encoding"* type=*"string"*></attribute>

</complexType>

<complexType name=*"EntitiesType"*>

<sequence>

<element name=*"Entity"* type=*"tns:EntityType"*

maxOccurs=*"unbounded"* minOccurs=*"1"*>

</element>

</sequence>

</complexType>

<complexType name=*"SignedDepositNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"* type=*"tns:DigitalPostmarkType"*

maxOccurs=*"1"* minOccurs=*"1"*>

</element>

<element name=*"EnvelopeInformation"*

type=*"tns:EnvelopeInformationType"* maxOccurs=*"1"* minOccurs=*"1"*>

</element>

</sequence>

</complexType>

<complexType name=*"DepositNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"* type=*"tns:DigitalPostmarkType"*

maxOccurs=*"1"* minOccurs=*"1"*>

</element>

</sequence>

</complexType>

<complexType name=*"TransitNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"* type=*"tns:DigitalPostmarkType"*

maxOccurs=*"1"* minOccurs=*"1"*>

</element>

</sequence>

</complexType>

<complexType name=*"SignedTransitNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"* type=*"tns:DigitalPostmarkType"*

maxOccurs=*"1"* minOccurs=*"1"*>

</element>

<element name=*"EnvelopeInformation"*

type=*"tns:EnvelopeInformationType"* maxOccurs=*"1"* minOccurs=*"1"*>

</element>

</sequence>

</complexType>

<complexType name=*"ReceiptNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"*

type=*"tns:DigitalPostmarkType"*>

</element>

<element name=*"EnvelopeInformation"*

type=*"tns:EntityEnvelopeInformationType"*>

</element>

</sequence>

</complexType>

<complexType name=*"SignedReceiptNoticeType"*>

<sequence>

<element name=*"DigitalPostmark"*

type=*"tns:DigitalPostmarkType"*>

</element>

<element name=*"EnvelopeInformation"*

type=*"tns:EntityEnvelopeInformationType"*>

</element>

</sequence>

</complexType>

<complexType name=*"HashValueType"*>

<attribute name=*"AlgorithmOID"*>

<simpleType>

<restriction base=*"string"*>

<enumeration value=*"1.3.14.3.2.26"*></enumeration>

<enumeration value=*"2.16.840.1.101.3.4.2.1"*></enumeration>

</restriction>

</simpleType>

</attribute>

</complexType>

<complexType name=*"EntityEnvelopeInformationType"*>

<sequence>

<element name=*"BodyEnvelopeInformation"* type=*"tns:ContentEnvelopeInformationType"*>

</element>

<element name=*"Entity"* type=*"tns:EntityType"*></element>

<element name=*"EntityChallenge"* type=*"tns:EntityChallengeType"*></element>

</sequence>

</complexType>

<complexType name=*"EntityChallengeType"*>

<sequence>

<element name=*"SecretQuestion"* type=*"tns:SecretQuestionType"*></element>

<element name=*"Signature"* type=*"ds:SignatureType"*></element>

</sequence>

</complexType>

<complexType name=*"RequestType"*>

<attribute name=*"RandomNumber"* type=*"string"*></attribute>

</complexType>

<complexType name=*"ResponseType"*>

<attribute name=*"AlgorithmIdentifier"* type=*"string"*></attribute>

</complexType>

</schema>

Annex B  
  
Notices in ASN-1

(This annex forms an integral part of this Recommendation.)

To be updated from the XSD Schema

CMAIL DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS String

FROM XSDv2 {joint-iso-itu-t asn1(1) specification(0) modules(0)

xsd-module(2) version2(2)};

DepositNotice ::= DepositNoticeType

SignedDepositNotice ::= SignedDepositNoticeType

TransitNotice ::= TransitNoticeType

SignedTransitNotice ::= SignedTransitNoticeType

ReceiptNotice ::= ReceiptNoticeType

SignedReceiptNotice ::= SignedReceiptNoticeType

DigitalPostmarkType ::= SEQUENCE {

mimeMessageHash SEQUENCE (SIZE(1..MAX)) OF

mimeMessageHash HashValueType,

signature SEQUENCE (SIZE(0..MAX)) OF

signature SignatureType,

envelopeId String,

deliveryType ENUMERATED {

certifiedMail,

...

}

}

EnvelopeInformationType ::= SEQUENCE {

contentEnvelopeInformation ContentEnvelopeInformationType,

entities EntitiesType,

signature SEQUENCE (SIZE(0..MAX)) OF

signature SignatureType

}

ContentEnvelopeInformationType ::= SEQUENCE {

uncipheredEnvelopeHash HashValueType,

cipheredEnvelopeHash HashValueType,

messageId String

}

SecretQuestionType ::= SEQUENCE {

request RequestType,

response ResponseType

}

EntityType ::= SEQUENCE {

secretQuestion SecretQuestionType,

cipheredEnvelopeKey CipheredEnvelopeKeyType,

certificate CertificateType,

emailAddress String

(CONSTRAINED BY

{-- "Email address has to be in RFC 822 format --}),

type ENUMERATED {

from,

to,

cc,

transit

}

}

CipheredEnvelopeKeyType ::= SEQUENCE {

algorithm String,

cipherededKey String,

encoding String,

keySize String

}

CertificateType ::= SEQUENCE {

encoding String

}

EntitiesType ::= SEQUENCE {

entity SEQUENCE(SIZE(1..MAX)) OF entity EntityType

}

SignedDepositNoticeType ::= SEQUENCE {

digitalPostmark DigitalPostmarkType,

envelopeInformation EnvelopeInformationType

}

DepositNoticeType ::= SEQUENCE {

digitalPostmark DigitalPostmarkType

}

TransitNoticeType ::= SEQUENCE {

digitalPostmark DigitalPostmarkType

}

SignedTransitNoticeType ::= SEQUENCE {

digitalPostmark DigitalPostmarkType,

envelopeInformation EnvelopeInformationType

}

ReceiptNoticeType ::= SEQUENCE {

operatorPostmark DigitalPostmarkType

}

SignedReceiptNoticeType ::= SEQUENCE {

operatorPostmark DigitalPostmarkType,

envelopeInformation EntityEnvelopeInformationType

}

HashValueType ::= SEQUENCE {

algorithmOID ENUMERATED {

sha-1,

sha-256

}

}

EntityEnvelopeInformationType ::= SEQUENCE {

bodyEnvelopeInformation ContentEnvelopeInformationType,

entity EntityType,

entityChallenge EntityChallengeType

}

EntityChallengeType ::= SEQUENCE {

secretQuestion \_SecretQuestionType,

signature SignatureType

}

RequestType ::= SEQUENCE {

randomNumer String

}

ResponseType ::= SEQUENCE {

algorithmIdentifier String

}

SignatureType ::= String

ENCODING-CONTROL XER

GLOBAL-DEFAULTS MODIFIED-ENCODINGS

[NAME AS CAPITALIZED] DigitalPostmarkType.mimeMessageHash

[UNTAGGED] DigitalPostmarkType.mimeMessageHash

[NAME AS CAPITALIZED] DigitalPostmarkType.signature.\*

[UNTAGGED] DigitalPostmarkType.signature

[NAME AS CAPITALIZED] DigitalPostmarkType.envelopeId

[ATTRIBUTE] DigitalPostmarkType.envelopeId

[NAME AS CAPITALIZED] DigitalPostmarkType.deliveryType

[ATTRIBUTE] DigitalPostmarkType.deliveryType

[TEXT AS CAPITALIZED] DigitalPostmarkType.delivetyType:certifiedMail

[NAME AS CAPITALIZED] EnvelopeInformationType.contentEnvelopeInformation

[NAME AS CAPITALIZED] EnvelopeInformationType.entities

[NAME AS CAPITALIZED] EnvelopeInformationType.signature

[UNTAGGED] EnvelopeInformationType.signature

[NAME AS CAPITALIZED]

ContentEnvelopeInformationType.uncipheredEnvelopeHash

[NAME AS CAPITALIZED]

ContentEnvelopeInformationType.cipheredEnvelopeHash

[NAME AS CAPITALIZED] ContentEnvelopeInformationType.messageId

[ATTRIBUTE] ContentEnvelopeInformationType.messageId

[NAME AS CAPITALIZED] SecretQuestionType.request

[NAME AS CAPITALIZED] SecretQuestionType.response

[NAME AS CAPITALIZED] EntityType.secretQuestion

[NAME AS CAPITALIZED] EntityType.cipheredEnvelopeKey

[NAME AS CAPITALIZED] EntityType.certificate

[NAME AS CAPITALIZED] EntityType.emailAddress

[ATTRIBUTE] EntityType.emailAddress

[NAME AS CAPITALIZED] EntityType.type

[ATTRIBUTE] EntityType.type

[NAME AS CAPITALIZED] CipheredEnvelopeKeyType.algorithm

[ATTRIBUTE] CipheredEnvelopeKeyType.algorithm

[NAME AS CAPITALIZED] CipheredEnvelopeKeyType.cipheredKey

[ATTRIBUTE] CipheredEnvelopeKeyType.cipheredKey

[NAME AS CAPITALIZED] CipheredEnvelopeKeyType.encoding

[ATTRIBUTE] CipheredEnvelopeKeyType.encoding

[NAME AS CAPITALIZED] CipheredEnvelopeKeyType.keysize

[ATTRIBUTE] CipheredEnvelopeKeyType.keysize

[NAME AS CAPITALIZED] CertificateType.encoding

[ATTRIBUTE] CertificateType.encoding

[UNTAGGED] EntitiesType.entity

[NAME AS CAPITALIZED] EntitiesType.entity.\*

[NAME AS CAPITALIZED] SignedDepositNoticeType.digitalPostmark

[NAME AS CAPITALIZED] SignedDepositNoticeType.envelopeInformation

[NAME AS CAPITALIZED] DepositNoticeType.digitalPostmark

[NAME AS CAPITALIZED] TransitNoticeType.digitalPostmark

[NAME AS CAPITALIZED] SignedTransitNoticeType.digitalPostmark

[NAME AS CAPITALIZED] SignedTransitNoticeType.envelopeInformation

[NAME AS CAPITALIZED] ReceiptNoticeType.digitalPostmark

[NAME AS CAPITALIZED] SignedReceiptNoticeType.digitalPostmark

[NAME AS CAPITALIZED] SignedReceiptNoticeType.envelopeInformation

[NAME AS CAPITALIZED] HashValueType.algorithmOID

[ATTRIBUTE] HashValueType.algorithmOID

[TEXT AS "1.3.14.3.2.26"] HashValueType.algorithmOID:sha-1

[TEXT AS "2.16.840.1.101.3.4.2.1"] HashValueType.algorithmOID:sha-256

[NAME AS CAPITALIZED]

EntityEnvelopeInformationType.BodyEnvelopeInformation

[NAME AS CAPITALIZED]

EntityEnvelopeInformationType.entityChallenge

[NAME AS CAPITALIZED] EntityChallengeType.secretQuestion

[NAME AS CAPITALIZED] EntityChallengeType.signature

[NAME AS CAPITALIZED] RequestType.randomNumber

[ATTRIBUTE] RequestType.randomNumber

[NAME AS CAPITALIZED] ResponseType.algorithmIdentifier

[ATTRIBUTE] ResponseType.algorithmIdentifier

END

Annex C  
  
Requirements on public-infrastructure components

(This annex forms an integral part of this Recommendation.)

## C.1 Cmail server end-entity public-key certificates

An end-entity public-key certificate issued to a Cmail server shall have the following content:

a) The version 3 shall be specified.

b) The CA shall generate non-sequential serial numbers.

c) The issuer field shall consist of a directory distinguished name consisting of two components:

‑ The first component shall be an attribute of type country. The value shall be the two-letter ISO 3166-1 country code for the country in which the issuer’s place of business is located.

‑ The second component shall be an attribute of type organizationName and the value field shall contain the name (or abbreviation thereof), trademark, or other meaningful identifier for the CA, provided that they accurately identify the CA within the country. The value shall not contain a generic designation such as “Root” or “CA1”.

‑Editor comment: to be reviewed

d) The subject field shall hold a directory distinguished name with components as follows:

‑ countryName. It shall be present and shall hold the two-letter ISO 3166-1 country code.

‑ organizationName: It shall be present and shall hold the legal name of the organisation managing the Cmail server;

‑ streetAddress: It shall be present and shall hold the street name and the house number;

‑ localityName: It shall be present and shall hold the name of the locality;

‑ stateOrProvinceName: It shall be present if necessary for unique identification. Otherwise is shall be absent.

‑ postalCode: It shall be present and shall hold the postal code for the location.

‑Editor comment: to be reviewed

e) The Subject alternative name extension shall be present. It shall contain two names in the sequence indicated below:

‑ Either dNSName or iPAddress shall be specified. The dNSName shall hold the fully qualified domain name for the server, while the iPAddress shall hold the IP address of the server. The IP address shall not be a reserved IP address.

‑ rfc822Name shall hold the e-mail address of administrator of the Cmail server.

f) The certificatePolicies extension shall be present and shall at least hold the object identifier {itu-t(0) recommendation(0) x(24) cmail(xxxx) certificateProfile(2) cmailServer(1)} to signal that the public-key certificate is issued according to this Recommendation.

Editor comment: to be reviewed

## C.2 Cmail client end-entity public-key certificates

An end-entity public-key certificate issued to a Cmail client shall have the following content:

a) The version 3 shall be specified.

b) The CA shall generate non-sequential serial numbers.

c) The issuer field shall consist of a director distinguished name consisting of two components:

‑ The first component shall be an attribute of type organizationName and the value field shall contain the name (or abbreviation thereof), trademark, or other meaningful identifier for the CA, provided that they accurately identify the CA within the country. The value shall not contain a generic designation such as “Root” or “CA1”.

‑ The second component shall be an attribute of type country. The value shall be the three-letter code (alpha-3) of ISO 3166-1 for the country in which the issuer’s place of business is located.

Editor comment: to be reviewed

d) The subject field shall hold a directory distinguished name with components as follows:

‑ surname shall be present if the client is an individual, but shall be absent if the client is an organisation.

‑ givenName shall be present if the surname is present, otherwise it shall be absent.

‑ initials may be present if surname is present. Otherwise, it shall be absent.

‑ generationQualifier may be present if surname is present. Otherwise, it shall be absent.

‑ organizationName shall be present if the client is not a residential person. Otherwise it shall be absent. If present, it shall hold the legal name of the organisation to whic the client belongs.

‑ streetAddress shall be present and shall hold the street name and the house number;

‑ localityName shall be present and shall hold the name of the locality;

‑ stateOrProvinceName shall be present if necessary for unique identification. Otherwise is shall be absent.

‑ postalCode shall be present and shall hold the postal code for the location.

‑ countryCode3 shall be present and shall hold the three-letter code (alpha-3) of ISO 3166-1.

Editor comment: to be reviewed

e) The Subject alternative name extension shall be present. It shall contain one element as indicated below:

‑ rfc822Name shall hold the e-mail address of administrator of the Cmail server.

f) The certificatePolicies extension shall be present and shall at least hold the object identifier {itu-t(0) recommendation(0) x(24) cmail(xxxx) certificateProfile(2) cmailClient(2)} to signal that the public-key certificate is issued according to this Recommendation.

Editor comment: to be reviewed

## C.3 Trust anchor CA-certificates

The trust anchor CA-certificate should not contain the certificatePolicies extension.

A Cmail server shall in its trust anchor storage have trust anchor CA- certificates for all other Cmail servers, with which it is communicating.

## C.4 Subordinate CA-certificates

A CA located between the trust anchor and the end-entity shall:

‑ if the CA is not affiliated with the issuing CA (or trust anchor), it shall in its certificatePolicies extension having one or both of the policy object identifiers listed in C.1 and C.2 depending on what kind of public-key certificates it issues. It shall not contain the anyPolicy identifier (2.5.29.32.0).

‑ if the CA is affiliated with the issuing CA, it may contain the anyPolicy identifier (2.5.29.32.0).

## C.5 Information validation requirements

Editor comment: to be provided

Annex D  
  
Requirements on Transport Layer Security (TLS)

(This annex forms an integral part of this Recommendation.)

TLS 1.2or later shall be supported.

In the negotiation, neither the Cmail server nor the client shall accept a connection where there is an attempt negotiate a TLS version earlier than TLS 1.2.

An implementation shall support the following cipher suite:

‑ TLS\_DH\_RSA\_WITH\_AES\_256\_CBC\_SHA256

Editor's note: There are some known attacks on TLS in HTTP (Web service) environment. It has to be analyses how such attacks can mitigated in a non-HTTP environment.

Appendix I  
  
<Appendix Title>

(This appendix does not form an integral part of this Recommendation.)

<Body of appendix I>

Bibliography

**Object Identifier (OID)**

[ITU-T X.660], Recommendation ITU-T X.660 (2011) | ISO/IEC 9834-1:2012 Information technology - Open systems interconnection - Procedures for the operation of OSI registration authorities: General procedures and top arcs of the ASN.1 object identifier tree

Editor’s note: Remove some specifications

**Simple Mail Transfer Protocol (SMTP)**

[RFC 1870], SMTP Service Extension for Message Size Declaration (obsoletes: RFC 1653)

[RFC 3030], SMTP Service Extensions for Transmission of Large and Binary MIME Messages

[RFC 3207], SMTP Service Extension for Secure SMTP over Transport Layer Security (obsoletes RFC 2487)

[RFC 3463], Enhanced Status Codes for SMTP (obsoletes RFC 1893)

[RFC 3464], An Extensible Message Format for Delivery Status Notifications (obsoletes RFC 1894)

[RFC 3798], Message Disposition Notification

[RFC 4954], SMTP Service Extension for Authentication (obsoletes RFC 2554)

[RFC 5322], Internet Message Format (obsoletes RFC 822 aka STD 11 and RFC 2822)

[RFC 5336], SMTP Extension for Internationalized Email Addresses (updates RFC 2821, RFC 2822, and RFC 4952)

[RFC 6522], The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages (obsoletes RFC 3462, and in turn RFC 1892)

Editor’s note : Remove some specifications

**Post Office Protocol (POP3)**

[RFC 1939], Post Office Protocol – Version 3 (STD 53)

[RFC 2195], IMAP/POP AUTHorize Extension for Simple Challenge/Response

[RFC 2449], POP3 Extension Mechanism

[RFC 2595], Using TLS with IMAP, POP3 and ACAP

[RFC 5034], The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism

**Transport Layer Security (TLS)**

[RFC 5288], “AES Galois Counter Mode (GCM) Cipher Suites for TLS”.

[RFC 5289], “TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)”.

[RFC 5746], “Transport Layer Security (TLS) Renegotiation Indication Extension”.

[RFC 5878], “Transport Layer Security (TLS) Authorization Extensions”.

[RFC 6066], “Transport Layer Security (TLS) Extensions: Extension Definitions”, includes Server Name Indication and OCSP stapling.

[RFC 6091], “Using OpenPGP Keys for Transport Layer Security (TLS) Authentication“.

[RFC 6176], “Prohibiting Secure Sockets Layer (SSL) Version 2.0”.

**Domain Key Identified Mail (DKIM)**

[RFC 4871], DomainKeys Identified Mail (DKIM) Signatures Proposed Standard

[RFC 5617], DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP)

[RFC 6376], DomainKeys Identified Mail (DKIM) Signatures Draft Standard

[RFC 6377], DomainKeys Identified Mail (DKIM) and Mailing Lists

**Multipurpose Internet Mail Extensions (MIME)**

MIME is specified in six linked RFC memoranda: RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 and RFC 2049, which together define the specifications.

**Secure Multipurpose Internet Mail Extensions (S/MIME)**

[RFC 2045], Multipurpose Internet Mail Extensions (MIME) Part One was published in November 1996

[RFC 5751], Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification

[RFC 3156], MIME Security with OpenPGP

IETF RFC 5750, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2, Certificate Handling

**XML Advanced Electronic Signatures (XAdES)**

W3C XAdES version 1.1.1 from 2003 (http://www.w3.org/TR/XAdES/)

[ETSI TS 101 903], XAdES version 1.4.1 from 2009-06-15

[ETSI TS 101 903], V1.4.1 XSD

**XML Signature (XMLDsig)**

http://www.w3.org/TR/xmldsig-core/

\_\_\_\_\_\_\_\_\_\_