Examples of CMS Message Bodies

1. Introduction

The examples in this document show the structure and format of CMS message bodies, as described in [CMS]. They are useful to implementors who use protocols that rely on CMS, such as the S/MIME message format protocol [SMIME-MSG].

Every example in this document has been checked by two different implementors. This strongly indicates (but does not assure) that the examples are correct. All CMS implementors must read the CMS document carefully before implementing from it. No one should use the examples in this document as stand-alone explanations of how to create CMS message bodies.

This document explicitly does not attempt to cover many PKIX [PKIX] examples, nor does it cover any ESS [ESS] examples. Documents with examples of those formats may be forthcoming.

2. Contributions To This Document

The examples shown here were created and validated by many different people. In the example listings, there is a tag with the initials of the creator.
of the example, and one or more tags for the people who validated the example.

Some of the examples are of mis-implementations of CMS. That is, if a developer reading the CMS specification created a message body that was illegal, and another developer agreed that the mis-reading was potentially a pitfall for later developers, that message body is also included here. To make it clear which examples are bad, they are all put into a single section of this document with (hopefully) explicit headings.

To contribute an implementation of an unimplemented example listed in this document, to verify that you got the same results as an example listed here, or to suggest a new example that should be listed, please contact the document author at the address listed near the end of the document.

3. Constants Used in the Examples

This section defines the data used in the rest of the document. The names of the constants indicate their use. For example, AlicePrivDSSSign is the private part of Alice’s DSS signing key.

- Alice is the creator of the message bodies in this spec.
- Bob is the recipient of the messages.
- Carl is a CA.
- Diane sometimes gets involved with these folks.

3.1 Content of documents

ExContent is the following sentence:

    This is some sample content.

That is, it is the string of characters starting with "T" up to and including the ".".

The hex for ExContent is

5468 6973 2069 7320 736f 6d65 2073 616d 706c 6520 636f 6e74 656e 742e

The MD5 hash of ExContent is

9898 cac8 fab7 691f f89d c207 24e7 4a04

The SHA-1 hash of ExContent is

406a ec08 5279 ba6e 1602 2d9e 0629 c022 9687 dd48

3.2 Keys

The following keys are needed to create the samples. Note that BobPubDHEncrypt and DianePubDHEncrypt do *not* share Diffie-Hellman parameters. For the example of a common UKM, the BobPubDHSharedEncrypt and DianePubDHSharedEncrypt keys are defined in that example.

AlicePrivDSSSign = XXXXX
AlicePrivRSASign = XXXXX
AlicePubDSSSign = XXXXX
AlicePubRSASign = XXXXX
BobPrivDHEncrypt = XXXXX
BobPrivRSAEncrypt = XXXXX
BobPubDHEncrypt = XXXXX
3.3 Certificates

AliceDSSSignByCarl = XXXXX
AliceRSASignByCarl = XXXXX
BobDHEncryptByCarl = XXXXX
CarlDSSSelf = XXXXX
CarlRSASelf = XXXXX
DianeDSSSignByCarl = XXXXX
DianeRSASignByCarl = XXXXX

3.4 CRLs

CarlCRL is a CRL from Carl that contains three revocations.

CarlCRL = XXXXX

4. Trivial Examples

This section covers examples of small CMS types.

4.1 ContentInfo with Data type, BER

The object is a ContentInfo containing a Data object in BER format that is ExContent.

 XXXXX

4.1 ContentInfo with Data type, DER

The object is a ContentInfo containing a Data object in DER format that is ExContent.

DataTypeDER.bin:

 XXXXX

5. Signed-data

5.1 Basic signed content, DSS

A SignedData with no attribute certificates, signed by Alice using DH-DSS, just her certificate (not Carl’s root cert), no CRL. The message is ExContent, and is included in the eContent. There are no signed or unsigned attributes.

 XXXXX
5.2 Basic signed content, RSA

Same as 5.1, except using RSA signatures. A SignedData with no attribute certificates, signed by Alice using RSA, just her certificate (not Carl’s root cert), no CRL. The message is ExContent, and is included in the eContent. There are no signed or unsigned attributes.

5.3 Basic signed content, detached content

Same as 5.1, except with no eContent. A SignedData with no attribute certificates, signed by Alice using DH-DSS, just her certificate (not Carl’s root cert), no CRL. The message is ExContent, but the eContent is not included. There are no signed or unsigned attributes.

5.4 Fancier signed content

Same as 5.1, but includes Carl’s root cert, Carl’s CRL, some signed and unsigned attributes (Counter-signature by Diane). A SignedData with no attribute certificates, signed by Alice using DH-DSS, her certificate and Carl’s root cert, Carl’s DSS CRL. The message is ExContent, and is included in the eContent. The signed attributes are Content Type, Message Digest and Signing Time; the unsigned attributes are XXXXX.

5.6 All RSA signed message

Same as 5.2, but includes Carl’s RSA root cert (but no CRL). A SignedData with no attribute certificates, signed by Alice using RSA, her certificate and Carl’s root cert, no CRL. The message is ExContent, and is included in the eContent. There are no signed or unsigned attributes.

5.7 Multiple signers

Similar to 5.1, but the message is also signed by Diane. Two SignedDatas (one for Alice, one for Diane) with no attribute certificates, each signed using DH-DSS, Alice’s and Diane’s certificate (not Carl’s root cert), no CRL. The message is ExContent, and is included in the eContent. There are no signed or unsigned attributes.

5.8 Signing using SKI

Same as 5.1, but the signature uses the SKI instead of the issuer/serial number in the cert. A SignedData with no attribute certificates, signed by Alice using DH-DSS, just her certificate (not Carl’s root cert), identified by the SKI, no CRL. The message is ExContent, and is included in the eContent. There are no signed or unsigned attributes.

6. Enveloped-data
6.1 Basic encrypted content, TripleDES and DH

An EnvelopedData from Alice to Bob of ExContent using TripleDES for encrypting and Diffie-Hellman for key management. Does not have a OriginatorInfo or any attributes.

6.2 Basic encrypted content, TripleDES and RSA

Same as 6.1, except with RSA for key management. An EnvelopedData from Alice to Bob of ExContent using TripleDES for encrypting and RSA for key management. Does not have a OriginatorInfo or any attributes.

6.3 Basic encrypted content, RC2/40 and RSA

Same as 6.1, except using RC2/40 for encryption and RSA for key management. An EnvelopedData from Alice to Bob of ExContent using RC2/40 for encrypting and RSA for key management. Does not have a OriginatorInfo or any attributes.

6.4 Encrypted content, two recipients, no shared keying material

Same as 6.1, except sent to both Bob and Diane. An EnvelopedData from Alice to Bob and Diane of ExContent using TripleDES for encrypting and Diffie-Hellman for key management. Does not have a OriginatorInfo or any attributes.

6.5 Encrypted content, two recipients, shared keying material

Same as 6.4, except using keys that have shared parameters so the result does not include the UKMs. An EnvelopedData from Alice to Bob and Diane of ExContent using TripleDES for encrypting and Diffie-Hellman for key management. Does not have a OriginatorInfo or any attributes. Uses BobPubDHSharedEncrypt and DianePubDHSharedEncrypt for keys.

BobPubDHSharedEncrypt = XXXXX
DianePubDHSharedEncrypt = XXXXX

6.6 Encrypted content, TripleDES and DH, previously-distributed keys

Same as 6.1, except sent using a previously-distributed key. An EnvelopedData from Alice to Bob of ExContent using TripleDES for encrypting and Diffie-Hellman for key management, using the MailListTripleDES key. Does not have a OriginatorInfo or any attributes.

6.7 Encrypted content, RC2/40 and RSA, previously-distributed keys

Same as 6.1, except sent using a previously-distributed key. An EnvelopedData from Alice to Bob of ExContent using TripleDES for
encrypting and RSA for key management, using the MailListRC2 key. Does not have a OriginatorInfo or any attributes.

7. Digested-data

A DigestedData from Alice to Bob of ExContent using SHA-1.

8. Encrypted-data

An EncryptedData from Alice to Bob of ExContent with no attributes.

9. Authenticated-data

9.1 Authenticated data with no authenticated attributes

An AuthenticatedData from Alice to Bob using XXXXXXXXXX with no authenticated attributes.

9.2 Authenticated data with authenticated attributes

An AuthenticatedData from Alice to Bob using XXXXXXXXXX with the content-type and message-digest authenticated attributes.

A. References

[PKIX] PKIX Certificate and CRL Profile, RFC 2459.

B. Binaries of the Examples

This section contains the binaries of the examples shown in the rest of the document. The binaries are stored in a modified Base64 format. There is a Perl program that, when run over the contents of this document, will extract the following binaries and write them out to disk. The program works with Perl for Unix and Windows 95/98/NT (and possibly Macintosh).

B.1 How the binaries and extractor works

The program in the next section looks for lines that begin with a ‘|’ character (or some whitespace followed by a ‘|’), ignoring all other lines.
If the line begins with ‘|’, the second character tells what kind of line it is:

- * is a comment
- > gives the name of a new file to start
- < tells to end the file (and checks the file name for sanity)
- anythingelse is a Base64 line

The program writes out a series of files, so you should run this in an empty directory. The program will overwrite files (if it can), but won’t delete other files already in the directory.

Run this program with this document as the standard input, such as:

```
extractsample <draft-ietf-smime-examples
```

If you want to extract without the program, copy all the lines between the "|>" and "|<" markers, remove any page breaks, and remove the "|" in the first column of each line. The result is a valid Base64 blob that can be processed by any Base64 decoder.

### B.2 Example extraction program

```perl
#!/usr/bin/perl

# CMS Samples extraction program. v 1.1

# Get all the input as an array of lines
@AllIn = (); while (<STDIN>) { push(@AllIn, $_) }

$Base64Chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=';
$LineCount = 0; $CurrFile = '';

foreach $Line (@AllIn) {
    $LineCount++;  # Keep the line counter for error messages
    chomp($Line);  # Get rid of CR or CRLF at the end of the line
    if(substr($Line, 0, 1) ne '|') { next }  # Not a special line
    elsif(substr($Line, 1, 1) eq '*') { next }  # It is a comment
    elsif(substr($Line, 1, 1) eq '>')
        { &StartNewFile(substr($Line, 2)) }  # Start a new file
    elsif(substr($Line, 1, 1) eq '<')
        { &EndCurrFile(substr($Line, 2)) }  # End the current file
    else { &DoBase64(substr($Line, 1)) }  # It is a line of Base64
}

sub StartNewFile {
    $TheNewFile = shift(@_);
    if($CurrFile ne '') { die "Was about to start a new file at ".
        "line $LineCount, but the old file, $CurrFile, was open\n" )
    open(OUT, ">$TheNewFile") or
die "Could not open $TheNewFile for writing: $!");
    $CurrFile = $TheNewFile;
    $LeftOver = 0;  # Amount left from previous Base64 character
    $NextPos = 0;  # Bit position to start the next Base64 character
    # (bits are numbered 01234567)
    $OutString = '';  # Holds the text going out to the file
}

sub EndCurrFile {
    $FileToEnd = shift(@_);
    if($CurrFile ne $FileToEnd) { die "Was about to close ".
```
print OUT $OutString;
close(OUT);
$CurrFile = '';}

sub DoBase64 {
    $TheIn = shift(@_);
    if($CurrFile eq '') { die "Got some Base64 at line $LineCount, 
        "but appear to be writing to any particular file" }
    @Chars = split(/, $TheIn);  # Make an array of the characters
    foreach $ThisChar (@Chars) {
        # $ThisVal is the position in the string and the Base64 value
        $ThisVal = index($Base64Chars, $ThisChar);
        if($ThisVal == -1) { die "At line $LineCount, found the 
            "character $ThisChar, which is not a Base64 character
        } if($ThisVal == 64) { last }  # It is a ", so we’re done
        if ($NextPos == 0 ) {
            # Don’t output anything, just fill the left of $LeftOver
            $LeftOver = $ThisVal * 4;
            $NextPos = 6;
        } elsif ($NextPos == 2) {
            # Add $ThisVal to $LeftOver, output, and reset
            $OutString .= chr($LeftOver + $ThisVal);
            $LeftOver = 0;
            $NextPos = 0;
        } elsif ($NextPos == 4) {
            # Add upper 4 bits of $ThisVal to $LeftOver and output
            $Upper4 = ($ThisVal & 60);
            $OutString .= chr($LeftOver + ($Upper4/4));
            $LeftOver = (($ThisVal - $Upper4) * 64);
            $NextPos = 2;
        } elsif ($NextPos == 6) {
            # Add upper 2 bits of $ThisVal to $LeftOver and output
            $Upper2 = ($ThisVal & 48);
            $OutString .= chr($LeftOver + ($Upper2/16));
            $LeftOver = (($ThisVal - $Upper2) * 16);
            $NextPos = 4;
        } else { die "\$NextPos has an illegal value." }
    }
}

B.3 Examples by section

B.3.1 Examples from section 3.1

[* ExContent is just the message; creator: [PH]
>ExContent.bin
VGhpcyBpcyBzb21lIHNhbXBsZSB2b250ZW50Lg==
<ExContent.bin

B.3.2 Examples from section 3.2

[* Here is the gang of keys
>AlicePrivDSSSign.key
blablahblablah
moreblablahblablah
<AlicePrivDSSSign.key
>AlicePrivRSASign.key
BlablahblahH
C. Acknowledgments

The following people contributed ideas and/or examples to this document. They are listed by their real names, with the initials used in the examples after their names.

Blake Ramsdell [BR]
Paul Hoffman [PH]
Jim Schaad [JS]

The examples are displayed with a modified version of Peter Gutmann’s "dumpasn1" program.

D. Editor’s Address

Paul Hoffman
Internet Mail Consortium
127 Segre Place
Santa Cruz, CA  95060 USA
phoffman@imc.org