Internet Draft
draft-ietf-cat-p7im-00.txt
Expires: November 30, 1996

C. Adams
NORTEL

May 30, 1996

PKCS #7-Based IDUP Mechanism (p7im)

STATUS OF THIS MEMO

ABSTRACT

The Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API) extends the GSS-API [RFC-1508] for applications requiring protection of a generic data unit (such as a file or message) in a way which is independent of the protection of any other data unit and independent of any concurrent contact with designated "receivers" of the data unit. Thus, it is suitable for applications such as secure electronic mail where data needs to be protected without any on-line connection with the intended recipient(s) of that data. Subsequent to being protected, the independent data unit can be transferred to the recipient(s) - or to an archive - perhaps to be processed only days or years later.

This document is a companion document to IDUP-GSS-API [IDUP] and IDUP: C-bindings [IDUP-C]. It provides a PKCS #7-based mechanism for IDUP (analogous to [Kerb5] or [SPKM] which provide underlying mechanisms for [GSS]). This mechanism specifies the procedures for creating and processing security formats according to that defined by the Public-Key Cryptography Standards set of documents (specifically Part 7 of the set [PKCS7], which specifies the "Cryptographic Message Syntax Standard"). Calling applications can use this IDUP mechanism to assist them in creating true S/MIME [S/MIME] objects. More precisely, a calling application is assumed by this mechanism to be MIME-aware and thus capable of performing all

necessary canonicalization and (base64 or quoted-printable) encoding on the data. The application thus uses the mechanism specified in this document exclusively for the security aspects of S/MIME.

1. INTRODUCTION

The Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API) [IDUP] provides security services to calling applications in a local environment. This PKCS #7-Based IDUP Mechanism (p7im) allows an application to "protect" an independent data unit (IDU) for future use and to "unprotect" a protected IDU, applying security services such as confidentiality, integrity and data origin authentication on a per-data-unit basis.

There are four stages to using the IDUP-GSS-API:

(a) The application acquires a set of credentials with which it may bind its identity to a data unit. The application's credentials vouch for its global identity, which may or may not be related to the local username under which it is running.

(b) The application establishes a security environment using its credentials. The security environment contains whatever information is necessary in order to provide per-IDU security services.

(c) Per-IDU calls are invoked to provide one of the following for PKCS #7-based IDU protection:
- data origin authentication with data integrity (SignedData);
- data confidentiality (EnvelopedData);
- both of the above (SignedAndEnvelopedData).

The application wishing to "protect" an IDU will call the protection set of IDUP-GSS-API routines, specifying the appropriate security environment. The recipient (wishing to "unprotect" the data) will pass the protected IDU (P-IDU) to the unprotection set of routines to remove the protection and/or to validate the data.

(d) At the completion of a security environment (which may extend across several protection and unprotection operations), the application calls an IDUP-GSS-API routine to abolish the security environment.

2. INDEPENDENT DATA UNIT PROTECTION MECHANISM

2.1. Protection Token

A P-IDU is a caller-opaque data structure that p7im uses to store protection information regarding an IDU. It is an OCTET STRING generated during the protection set of calls for use by p7im or by another mechanism during the unprotection set of calls. If encapsulation is requested by the calling application, the P-IDU consists entirely of the contents of the pidu_buffer, output_buffer, and final_pidu_buffer parameters used in the IDUP protection set of calls. Otherwise, the P-IDU consists of the contents of the midu_buffer, output_buffer, and final_midu_buffer parameters, along with the unencapsulated_token parameter.

2.2. Security Services

p7im provides the security services given in Section 1(c) above. The security services "proof of origin" and "service solicitation" (such as a request for "proof of delivery"), as defined in [IDUP], are not included in this PKCS #7-based IDUP mechanism. Receipt generation and processing are beyond the scope of [S/MIME] and other types of non-repudiable evidence generation and processing are not addressed in this version of p7im but may be included in future versions of this specification.

2.3. IDUP Parameter Bundle Uses and Defaults

The following parameter bundle uses and defaults are specified.

Mech_Specific_Info
- NOT USED (the only acceptable input, therefore, is NULL)

Idu_Sensitivity
- NOT USED (the only acceptable input, therefore, is NULL)

Service_Creation_Info
- NOT USED (the only acceptable input, therefore, is NULL)

Service_Verification_Info
- NOT USED (the only acceptable input, therefore, is NULL)

Quality
- the qop_algs parameter is supported. For p7im implementation and interoperability purposes, the following qop_algs values are defined.
  * For the Confidentiality MA field: 0001 (1) = RC2-40-CBC (this is also defined to be the TS "DEFAULT" conf. alg.);
    0002 (2) = DES-CBC;
    0003 (3) = DES-EDE3-CBC.
  * For the Integrity MA field: 0001 (1) = RSA-MD5 (this is also defined to be the TS "DEFAULT" int. alg.);

0002 (1) = RSA-MD2.

- the parameters validity, policy_id, and allow_policy_mapping are NOT USED (NULLs are therefore the only acceptable input)

Idu_Information

- the idu_type parameter must have a value representing a valid PKCS #7 content type (i.e., "Data", "SignedData", "EnvelopedData", "SignedAndEnvelopedData", "DigestedData, or "EncryptedData") or any other valid MIME type, including multipart (the DEFAULT value is specified to be "text/plain");
- the idu_title parameter is NOT USED (the only acceptable input, therefore, is NULL)

Prot_Information

- the idu_type (in Idu_Information) parameter is read from the S/MIME P-IDU and is output by IDUP_End_Unprotect()
- The originator_name parameter is read from the S/MIME P-IDU (the "signerInfos" (and possibly "certificates") field in the SignedData or SignedAndEnvelopedData content types) and is output by IDUP_End_Unprotect()
- all other parameters are NOT USED (and therefore NULL)

Special_Conditions

- NOT USED (the only acceptable input, therefore, is NULL)

Target_Info

- this bundle is used as described in IDUP; no DEFAULT values are specified

General_Service_Data

- the unencapsulated_token parameter is used only if encapsulation_request is FALSE);
- the minor_status parameter is used to return minor status values as specified in Section 5 below.

Prot_Service

- the prot_service_type parameter may have a value of "1" ("perform unsolicited service") or NULL (which specifies the DEFAULT value of "1");
- the service_id parameter must have a value representing "PER_CONF" or "PER_DOA";
- the parameters Service_Creation_Info, service_to, Service_Verification_Info, and service_verification_info_id are NOT USED (and therefore NULL)

Unprot_Service

- the unprot_service_type parameter will always have a value of "1" ("receive unsolicited service");
- the service_id parameter will have a value representing "REC_CONF" or "REC_DOA";
- the parameters service_verification_info_id, Service_Verification_Info, service_to, and Service_Creation_Info, are NOT USED (and therefore NULL)

3. SUMMARY OF TRANSFORMATIONS

The following composition of transformations is applied to the IDU for full S/MIME compliance during the IDUP protection set of calls (see [S/MIME] for discussions of Content-Transfer-Encoding and Canonicalization; "PerSecServ" refers to the performance of security services):

```
Transmit_Form = C-T-Encode(PerSecServ(Canonicalize(Local_Form)))
```

The inverse transformations are performed, in reverse order, to unprotect the IDU ("RemSecServ" refers to the removal/verification of security services):


```
Local_Form = DeCanonicalize(RemSecServ(C-T-Decode(Transmit_Form))).
```

It is the responsibility of the underlying p7im implementation to perform the PerSecServ and RemSecServ transformations above. The transformations Canonicalize, DeCanonicalize, C-T-Encode, and C-T-Decode are expected to be performed by the calling application unless one or more of the transformations is unnecessary (e.g., the input data is already in canonical form).

Note that the Local_Form and the functions to transform messages to and from Canonical_Form may vary between the protector and unprotector systems provided there is no loss of information.

4. TOKEN FORMAT

This section discusses protocol-visible characteristics of p7im; it defines elements of protocol for interoperability and is independent of any IDUP language bindings.

The p7im IDUP-GSS-API mechanism will be identified by an Object Identifier representing "p7im" , having the value:

```
{ iso(1) org(3) dod(5) internet(1) security(5) p7im(xx) }
```

The token transferred between IDUP-GSS-API peers (for IDU protection and unprotection purposes) is defined.

4.1. The Protected-IDU (P-IDU) Token

The Protected-IDU (from the output parameters of the protection set of calls) is a PKCS #7 ContentInfo, which itself is BER-encoded. The process for creating the P-IDU from the original input data (the IDU) consists of the steps described in [S/MIME], Section 4.4, after canonicalization and before base64 encoding (with all relevant ASN.1 specifications as described in [PKCS7]). Processing this token for the unprotection set of calls consists of the steps described in [S/MIME], Section 4.5 after base64 decoding and before de-canonicalization.

Thus, the input to the protection set of calls is treated as an opaque object (which will typically be a properly-encoded MIME object [RFC-1521]). This object is passed either in a single buffer to the IDUP_Start_Protect() call, or in multiple buffers (of arbitrary size) to the IDUP_Protect() call (one buffer per call). The result is a properly-encoded PKCS #7 ContentInfo, which may be content-transfer-encoded and placed within the body of an "application/x-pkcs7-mime" body part to produce a properly-encoded S/MIME object.

For unprotection, the input is assumed to be a properly-encoded S/MIME object (passed either as a single buffer to IDUP_Start_Unprotect() or as multiple buffers to IDUP_Unprotect()). The output is treated as an opaque object (but will typically be a properly-encoded MIME object, ready to be passed to a MIME parser for further processing.

4.2. Example of Protection Tokens

4.2.1 Signed Data

In similar fashion to the example given in [S/MIME], Section 4.4, assume that the MIME object (MO) to be protected is as follows:

```
Content-Type: text/plain; charset="us-ascii"

This is a signed message.
```

Using the IDUP protection set of calls with the p7im underlying mechanism yields the following sequence of events.

* MO is canonicalized to have <CR><LF> end-of-line delimiters (this step is performed by the calling application).

* Assuming a signing algorithm of md5-with-RSA, and assuming that certificates and CRLs do not need to be carried within the message in this particular environment, the canonicalized MIME object (CMO) is then signed according to PKCS #7 SignedData, so that the

resulting ASN.1 SEQUENCE has version=1, digestAlgorithms=md5, contentInfo=CMO, and signerInfos={version=1, issuerAndSerialNumber=..., digestAlgorithm=md5, digestEncryptionAlgorithm=RSA, encryptedDigest=34iur8a...834rnfz}.

* The PKCS #7 ContentInfo structure is then the ASN.1 SEQUENCE {contentType={pkcs-7 2}, content=SignedData}.

* ContentInfo is then BER-encoded.

* The BER-encoded ContentInfo is then base64-encoded (this step is performed by the calling application).

* The BER-encoded, base64-encoded ContentInfo becomes the body of an application/x-pkcs7-mime body part. The result might look like the following:

```
Content-Type: application/x-pkcs7-mime
Content-Transfer-Encoding: base64

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
```

(this step is performed by the calling application).

This would be sent in a MIME message to the intended recipient, who would reverse the above process (using the IDUP unprotection set of calls with underlying mechanism p7im, or any other S/MIME processor) to retrieve the original MIME object MO. This object can then be passed to the recipient's MIME parser, which will process it and display "This is a signed message." to the user.

5. MINOR STATUS CODES

No minor status codes have yet been defined for S/MIM.

6. SECURITY CONSIDERATIONS

Security issues are discussed throughout this memo.

7. REFERENCES

[IDUP] C. Adams, "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)", Internet Draft draft-ietf-cat-idup-gss.0x.txt (work in

progress).

[IDUP-C] D. Thakkar, D. Grebovich, "Independent Data Unit Protection Generic Security Service Application Program Interface: C-bindigs", Internet Draft draft-ietf-cat-idup-cbind-0x.txt (work in progress).

[KRB5] J. Linn, "The Kerberos Version 5 GSS-API Mechanism", Internet Draft draft-ietf-cat-kerb5gss-0x.txt (work in progress).

[PKCS7] RSA Laboratories, The Public-Key Cryptography Standards (PKCS) #7: "Cryptographic Message Syntax Standard", version 1.5, November 1, 1993.

[RFC-1508] J. Linn, "Generic Security Service Application Program Interface", RFC 1508.

[RFC-1521] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", September 1993.

[S/MIME] RSA Data Security, Inc., "S/MIME Message Specification: PKCS Security Services for MIME", Aug. 29, 1995.

[SPKM] C. Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)", Internet Draft draft-ietf-cat-spkmgss-0x.txt (work in progress).

8. AUTHOR'S ADDRESS

Carlisle Adams
NORTEL Secure Networks
P.O.Box 3511, Station C
Ottawa, Ontario, CANADA K1Y 4H7
Phone: (613) 763-9008
E-mail: cadams@bnr.ca