Network Working Group
INTERNET DRAFT
draft-ietf-pem-mime-07.txt

Steve Crocker
Ned Freed
Jim Galvin
Sandy Murphy
November 1994

PEM Security Services and MIME

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as ``work in progress''.

To learn the current status of any Internet Draft, please check the 1id-abstracts.txt listing contained in one of the Internet Drafts Shadow Directories on ds.internic.net (US East Coast), venera.isi.edu (US West Coast), munnari.oz.au (Pacific Rim), or nic.nordu.net (Europe).

Abstract

This document specifies how the services of MIME and PEM can be used in a complementary fashion. MIME, an acronym for "Multipurpose Internet Mail Extensions", defines the format of the contents of Internet mail messages and provides for multi-part textual and non-textual message bodies. PEM, an acronym for "Privacy Enhanced Mail", provides message authentication/integrity and message encryption services for Internet mail messages.

An Internet electronic mail message consists of two parts: the headers and the body. The headers form a collection of field/value pairs structured according to RFC822 [1], whilst the body, if structured, is defined according to MIME [2]. MIME does not provide for the application of security services.

PEM [3-6] specifies how to apply encryption and authentication/integrity services to the contents of a textual electronic mail message but does not provide message structuring or type labelling facilities. This document specifies how to use PEM with the multipart/signed and multipart/encrypted MIME content types to provide authentication/integrity and encryption services. We refer to the authentication/integrity service as a digital signature service.

This document specifies a number of changes to the message encryption and signature procedures of PEM and broadens the name forms that may be used to identify public keys. Many of the changes represent a departure in mechanism, not in effect.

1. Introduction

This document updates the message encryption and signature procedures defined by [3] and replaces the key certification and related services defined by [6]. The changes to [3] include the separation of the encryption and signature services, the removal of the limitation to enhance only text-based messages, the removal of the transfer encoding operation, the deprecation of the Content-Domain: and Proc-Type: headers, and the separation of certificate and certificate revocation list transmission from the security enhancements. These changes represent a departure in mechanism, not in effect, and are detailed in Section 8.

In addition, this document specifies four technical changes to PEM: symmetric key management in [3] is deprecated, the canonicalization operation in [3] is generalized, the allowable name forms for the identification of public keys is broadened to include arbitrary strings and email addresses, and users may distribute their public keys directly in lieu of certificates.

The key certification and related services are replaced by the specification of two new MIME content types: application/key-request and application/key-data. These new content types are used to transmit requests for key operations (storage, retrieval, certification, revocation list retrieval, etc.) and the responses to those requests. These two content types are independent body parts and are not required to be encapsulated in any other body part. These changes represent a departure in mechanism, not in effect, and are detailed in Section 8.

In order to make use of the PEM services, a user is required to have at

least one public/private key pair. Prior to this specification, the public key was required to be embodied in a certificate, an object that

binds a public key with a distinguished name, a name form that identified the owner of the public key. The embodiment was issued by a certification authority, a role that was expected to be trustworthy insofar as it verified the identity of the owner prior to issuing the certificate. However, the deployment of certificates and the creation of the hierarchy of certification authorities has been problematic.

Instead, this specification bases the PEM services on a public/private key pair. Each key pair is required to belong to a user (where user is not limited to being a human, e.g., it could be a process or a role) which has a name. There are 3 name forms specified by this document. For backward compatibility (and forward compatibility if the X.500 Directory becomes a ubiquitous service) one of the name forms is a distinguished name. In addition, email addresses and arbitrary strings are allowed.

Since a user may have more than one key pair, a name form is insufficient for uniquely identifying a key pair. A unique key selector must be assigned to each key pair. The combination of a name form and a key selector uniquely identifies a key pair and each key pair is uniquely identified by a name form and key selector combination. Throughout this document, this combination is called an identifier. There are 5 identifiers specified by this document.

> NOTE: In the simplest case, key selectors will be assigned by the owners of the key pairs. This works best when users generate their own key pairs for personal use, which they distribute to others asserting by declaration that the public key belongs to them. When the assertion that the public key belongs to them is made by a third party, for example when a certification authority issues a certificate to a user according to [4], the key selector may be assigned by that third party.

With a key pair for one's self and software that is both MIME and PEM aware, an originating user may digitally sign arbitrary data and send it to one or more recipients. With the public keys of the recipients, a

user may encrypt the data so that only the intended recipients can decrypt and read it. This specification separates these two services so that an originator may apply either or both, in either order.

The name forms and identifiers are described in detail in the next section. Succeeding sections specify how PEM and MIME are used together and other ancillary details.

2. Name Forms and Identifiers

Currently, [3] requires the use of certificates to create and receive PEM messages. Within certificates, [4] requires the use of distinguished names as specified by the X.500 Series of Recommendations. However, the Internet community has a great deal more experience with the use of electronic mail addresses as a name form and there is a desire to be able to use arbitrary strings to identify the owners of public keys. Hence, there is a need to support name forms which do not conform to the expected usage of distinguished names.

When receiving PEM messages it is necessary to be able to uniquely identify the key pair used to create the message. A certificate is one mechanism that accomplishes this, since it is uniquely identified by the combination of its issuer's distinguished name and its serial number. In any case, an identifier is required that consists of both a name form and key selector.

In addition, users may distribute their public keys via mechanisms outside the scope of the PEM specification, for example, in a file via FTP. Users receiving such keys will probably assign name forms to them to be displayed when receiving messages created with them. As a result, it is desirable to be able to explicitly specify the public key used rather than an identifier of the public key.

> NOTE: A feature of being able to specify the public key explicitly is that it allows users to exchange encrypted, anonymous mail. In particular, receiving users will always know a message comes from the same originating user.

The principal objective of the various Originator and Recipient fields specified in [3] is to identify which public key has been used or is required. This document reduces the set of fields by specifying exactly two: Originator-ID: for originators and Recipient-ID: for recipients. This specification defines 5 identifiers with which the public key used may be indicated in each of these fields.

In the next section the 3 name forms are described in detail. Following that is the specification of the 5 identifiers.

2.1. Name Forms

There are 3 name forms specified by this document: email addresses, distinguished names, and arbitrary strings.

2.1.1. Email Addresses

The email address (grammar token <emailstr>) used must be a valid RFC822 address, which is defined in terms of the two grammar tokens <addr-spec> and <route-addr>. The grammar for these two tokens is included in the Appendix as a convenience; the definitive source for these tokens is necessarily RFC822 [1].

```
    <emailstr>      ::= <addr-spec> / <route-addr>
                        ; an electronic mail address as defined by
                        ; these two tokens from RFC822
```

For example, the strings "crocker@tis.com", "galvin@tis.com", "murphy@tis.com", and "ned@innosoft.com" are all email addresses.

2.1.2. Arbitrary Strings

The arbitrary string (grammar token <string>) must have a length of at least 1. There are no other restrictions on the value chosen.

```
    <string>        ::= ; a non-null sequence of characters
```

For example, the string

```
    Jim "the SAAG mailing list maintainer" Galvin
```

is an arbitrary string.

2.1.3. Distinguished Names

The distinguished name (grammar token <dnamestr>) must be constructed according to the guidelines of the X.500 Directory. The actual syntax of the distinguished name is outside the scope of this specification. However, RFC1422, for example, specifies syntactic restrictions based on a particular choice of a certification hierarchy for certificates.

For the purposes of conveying a distinguished name from an originator to a recipient, it must be ASN.1 encoded and then printably encoded according to the base64 encoding defined by MIME.

```
    <dnamestr>      ::= <encbin>
                        ; a printably encoded, ASN.1 encoded
                        ; distinguished name (as defined by the 'Name'
                        ; production specified in X.501)
```


For example,

```
    /Country Name=US
    /State or Province Name=MD
    /Organization Name=Trusted Information Systems
    /Organizational Unit Name=Glenwood
    /Common Name=James M. Galvin/
```

is a distinguished name in a user friendly format (line breaks and leading spaces present only to improve readability). When encoded, it would appear as follows (line breaks present only to improve readability):

```
    MG0xCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNRDEkMCIGA1UEChMbVHJ1c3RlZCBJ
    bmZvcm1hdGlvbiBTeXN0ZW1zMREwDwYDVQQLEwhHbGVud29vZDEYMBYGA1UEAxMP
    SmFtZXMgTS4gR2Fsdmlu
```

2.2. Identifiers

There are 5 types of identifiers specified by this document: email address identifiers, arbitrary string identifiers, distinguished name identifiers, the public keys themselves, and issuer name serial number pairs from a certificate. All of these have approximately the same structure (except issuer name and serial number which has 'TYPE, STRING, KEYSEL' for historical reasons):

```
    TYPE, KEYSEL, STRING
```

The TYPE field is a literal string, one for each of the possible identifiers.

The KEYSEL field is used to distinguish between the multiple public keys that may be associated with the name form in the STRING field. Its value must be distinct from all other KEYSELs assigned by whomever assigned this KEYSEL. A suggested value is to use a portion (low-order 16 or 32 bits) or all of the actual public key used.

The STRING field is the name form and has a different syntax according to the value of the TYPE field.

The identifier used in each of the originator and recipient fields is described by the following grammar. The definition of the key selector token is included here since it used by several of the identifiers

below.

```
    <id>            ::=   <id-email> / <id-string>    / <id-dname>
                        / <id-publickey> / <id-issuer>

    <keysel>        ::= <encbin>
                        ; a printably encoded non-null sequence of octets
```

Each of the identifier name forms is described below.

2.2.1. Email Address

The email address identifier has the following syntax.

```
    <id-email>      ::= "EN"  "," <keysel> "," <emailstr> CRLF
```

The syntax of the token <emailstr> is defined in Section 2.1.1.

2.2.2. Arbitrary String

The arbitrary string identifier has the following syntax.

```
    <id-string>     ::= "STR" "," <keysel> "," <string> CRLF
```

The syntax of the token <string> is defined in Section 2.1.2.

2.2.3. Distinguished Name

The distinguished name identifier has the following syntax.

```
    <id-dname>      ::= "DN"  "," <keysel> "," <dnamestr> CRLF
```

The syntax of the token <dnamestr> is defined in Section 2.1.3.

2.2.4. Public Key

The public key identifier has the following syntax.


```
    <id-publickey>  ::= "PK"  "," <publickey> [ "," <id-subset> ] CRLF

    <publickey>     ::= <encbin>
                        ; a printably encoded, ASN.1 encoded public key (as
                        ; defined by the 'SubjectPublicKeyInfo' production
                        ; specified in X.509)
```

```
    <id-subset>     ::= <id-email> / <id-string> / <id-dname>
```

The object subjectPublicKeyInfo is imported from the X.500 Directory from the certificate object. It is currently the best choice for a general purpose public key encoding.

In normal usage, the token <id-subset> is expected to be absent. When present, it represents a mechanism by which an identifier (name form and key selector) can be associated with a public key. Recipients of a public key identifier must take care to verify the accuracy of the purported association. If not, it may be possible for a malicious originator to assert an identifier that accords the originator unauthorized privileges. See Section 5.2 for more details.

2.2.5. Issuer Name and Serial Number

The issuer name and serial number identifier has the following syntax.

```
    <id-issuer>     ::= "IS"  "," <dnamestr>  "," <serial> CRLF

    <serial>        ::= 1*<hexchar>
                        ; hex dump of the serial number of a certificate
```

The <id-issuer> identifier is included for backward compatibility (and forward compatibility if X.500 Directory certificates become ubiquitously available) with the ID-ASymmetric fields defined in [3]. Its syntax was chosen such that the older fields are easily converted to this new form by prefixing the old value with "IS," and replacing the field name with an appropriate new ID field name.

3. Applying PEM Security Services to MIME Body Parts

The next section describes the processing steps necessary to prepare a MIME body part for the application of PEM security services. The succeeding two sections describe the content of the multipart/signed and multipart/encrypted body parts resulting from the application of PEM

security services to MIME body parts.

3.1. PEM Processing Steps

The definition of the multipart/signed and multipart/encrypted body parts in [7] specifies three steps for creating both body parts.

(1) The body part to be protected is created according to a local convention.

(2) The body part is prepared for protection according to the protocol parameter.

(3) The prepared body part is protected according to the protocol parameter.

This specification makes no changes to step one in the sequence. For step two, there is no preparation necessary for the encryption service. For the digital signature service, the body part must be canonicalized as described below. This specification makes no changes to step three in the sequence.

Prior to the application of the digital signature service, the body part must be in a canonical form. Transforming the body part to be signed into a canonical form is a necessary and essential step in the digital signature process. The canonical form must satisfy the property that it is uniquely and unambiguously representable in both the originator and recipient's local environment. This is required in order to ensure that both the originator and recipient have the same data with which to calculate the digital signature; the originator needs to be able to include the digital signature value when transferring the body part, while the recipient needs to be able to compare a re-computed value with the received value. Further, the canonical form should satisfy the property that it is representable on as many different host computers as possible. By satisfying this property, signed data may be forwarded by recipients to additional recipients, who will also be able to verify the original signature. This service is called forwardable authentication.

The canonicalization transformation is a two step process. First, the body part must be converted to a form that is uniquely and unambiguously representable on as many different host computers as possible. Second, the body part must have its line delimiters converted to a unique and unambigous representation prior to computing the digital signature and prior to each verification of the digital signature.

The representation of all body parts in the first step is specified to be 7bit, as defined by [2]. Since the headers of body parts are already required to be representable in 7bit, this step requires that if the data to be signed is not already 7bit then it must be encoded with an appropriate MIME content transfer encoding. Note: since the MIME standard explicitly disallows nested content transfer encodings, i.e., the content types multipart and message may not themselves be encoded, body parts enclosed within, for example, a multipart content type must be encoded in a 7bit representation. Any valid MIME encoding may be selected for encoding the content of each of the non-7bit body parts.

As may be required by MIME, an appropriate Content-Transfer-Encoding: header is added and included with the data to be signed. Upon receipt, a MIME implementation would verify the signature of the data prior to decoding the data and displaying it to the recipient.

Representing all complex content types as 7bit transforms them into text-based content types. However, text-based content types present a unique problem. In particular, the line delimiter used for a text-based content type is specific to a local environment; different environments use the single character carriage-return (<CR>), the single character line-feed (<LF>), or the two character sequence "carriage-return line-feed (<CR><LF>)".

The application of the digital signature service requires that the same line delimiter be used by both the originator and the recipient. This document specifies that the two character sequence "<CR><LF>" must be used as the line delimiter. Thus, the second step of the canonicalization transformation includes the conversion of the local line delimiter to the two character sequence "<CR><LF>".

The conversion to the canonical line delimiter is only required for the purposes of computing the digital signature. Thus, originators must apply the line delimiter conversion before computing the digital signature but must transfer the data without the line delimiter conversion. Similarly, recipients must apply the line delimiter conversion before computing the digital signature.

> NOTE: An originator can not transfer the content with the line delimiter conversion intact because the conversion process is not idempotent. In particular, SMTP servers may themselves convert the line delimiter to a local line delimiter, prior to the message being delivered to the user. Thus, a recipient has no way of knowing if the conversion is present or not. Thus, if the recipient applies the conversion to a content in which it is

already present, the resulting content may have two line delimiters present, which would cause the verification of the signature to fail.

IMPLEMENTORS NOTE: Implementors should be aware that the conversion to a 7bit representation is a function that is available in a minimally compliant MIME user agent. Further, the line delimiter conversion required here is distinct from the same conversion included in that function. Specifically, the line delimiter conversion applied when a body part is converted to a 7bit representation is performed prior to application of the transfer encoding. The line delimiter conversion applied when a body part is signed is performed after the body is converted to 7bit.

3.2. Use of multipart/signed Content Type

Using this content type requires the specification of a control information content type, the label of which is used as the value for the required protocol parameter. Section 3.4 defines the control information content type of application/pem-signature. The value of the required parameter "protocol" is "application/pem-signature" and the value of the required parameter "micalg" is one of the valid choices from [5], for example:

```
Content-Type: multipart/signed; protocol="application/pem-signature";
  micalg="rsa-md5"; boundary="Signed Message"

--Signed Message
Content-Type: text/plain

This is some example text.

--Signed Message
Content-Type: application/pem-signature

SIGNATURE INFORMATION
--Signed Message--
```

where SIGNATURE INFORMATION is descriptive of the content that would appear in a real body part.

3.3. Use of multipart/encrypted Content Type

Using this content type requires the specification of a control information content type, the label of which is used as the value for the required protocol parameter. Section 3.5 defines the control information content type of application/pem-keys. The value of the required parameter "protocol" is "application/pem-keys", for example:

```
Content-Type: multipart/encrypted; protocol="application/pem-keys";
  boundary="Encrypted Message"

--Encrypted Message
Content-Type: application/pem-keys

KEY DATA

--Encrypted Message
Content-Type: application/octet-stream

ENCRYPTED DATA WOULD BE HERE
--Encrypted Message--
```

3.4. application/pem-signature Content Type Definition

(1) MIME type name: application

(2) MIME subtype name: pem-signature

(3) Required parameters: none

(4) Optional parameters: none

(5) Encoding considerations: quoted-printable is always sufficient

(6) Security considerations: none

This content type is used on the second body part of an enclosing multipart/signed when the protocol used is PEM. It is comprised of the digital signature of the data, which is the first body part of the enclosing multipart/signed, and the information required to verify that signature. The label application/pem-signature is used as the value of the protocol parameter of the enclosing multipart/signed. It is an error for the protocol parameter to be missing from the enclosing multipart/signed body part or for its value to be different from

application/pem-signature when this body part is used.

Included in the signature verification information will be the Message Integrity Check (MIC) algorithm used during the signature creation process. The MIC algorithm identified in this body part must match the MIC algorithm identified in the micalg parameter of the enclosing multipart/signed. If it does not, a user agent should identify the discrepancy to a user and may choose to either halt or continue processing, giving precedence to the algorithm identified in this body part.

An application/pem-signature body part is constructed as follows:

```
    Content-Type: application/pem-signature

    <pemsig>
```

where the <pemsig> token is defined as follows.

```
    <pemsig>        ::= <version> ( 1*<origasymflds> )

    <version>       ::= "Version:" "5" CRLF

    <origasymflds>  ::= <origid> <micinfo>

    <origid>        ::= "Originator-ID:" <id> CRLF
```

The token <id> is defined in Section 2.2.

3.5. application/pem-keys Content Type Definition

(1) MIME type name: application

(2) MIME subtype name: pem-keys

(3) Required parameters: none

(4) Optional parameters: none

(5) Encoding considerations: quoted-printable is always sufficient

(6) Security considerations: none

This content type is used on the first body part of an enclosing multipart/encrypted. It is comprised of the data encryption key used to encrypt the data, located in the second body part of the enclosing multipart/encrypted, and the information required to perform the decryption. The label application/pem-keys is used as the value of the protocol parameter of the enclosing multipart/encrypted. It is an error for the protocol parameter to be missing in the enclosing multipart/encrypted body part or for its value to be different from application/pem-keys when this body part is used.

An application/pem-keys body part is constructed as follows:

```
Content-Type: application/pem-keys

<pemkeys>
```

where the <pemkeys> token is defined as follows.

```
<pemkeys>       ::= <version> <dekinfo> 1*<recipasymflds>

<version>       ::= "Version:" "5" CRLF
```

```
    <recipasymflds> ::= <recipid> <asymkeyinfo>

    <recipid>       ::= "Recipient-ID:" <id> CRLF

    <asymkeyinfo>   ::= "Key-Info" ":" <ikalgid> "," <asymencdek> CRLF
```

The token <id> is defined in Section 2.2.

4. Removing PEM Security Services from PEM Body Parts

This section describes the processing steps necessary to verify or decrypt the PEM security services that have been applied to MIME body parts. Outer layers of PEM security services must be processed prior to processing inner layers of PEM security services. Processing includes a user choosing to display a content without removing the PEM security services.

The definition of the multipart/signed and multipart/encrypted body parts in [7] specifies three steps for receiving both body parts.

(1) The protected body part and the control information body part are prepared for processing.

(2) The prepared body parts are made available to the protection removal process.

(3) The results of the protection removal process are made available to the user and processing continues with the unprotected body part, as returned by the protection removal process.

For step one, the preparation for digitally signed and encrypted body parts is different, as described below. No changes are required to steps two and three in the sequence.

For multipart/signed body parts, the control information is prepared by removing any content transfer encodings that may be present. The digitally signed body part is prepared by leaving the content transfer

encodings intact and converting the line delimiters according to Step 2 of Section 3.1.

Multipart/encrypted body parts are prepared by removing the content transfer encodings, if present, from both the control information and the encrypted body part.

5. Key Management Content Types

This document defines two key management content types, the contents of which comprise a replacement mechanism for those defined in [6]. The first content type is application/pemkey-request, which replaces the certification and CRL-retrieval request messages. The second content type is application/pemkey-data, which replaces the certification reply message, the crl-storage request message, and the crl-retrieval reply message. There are no requirements for a crl-storage reply message and none are specified in this document. This document includes a specification for a public key and certificate request message, which were previously undefined.

5.1. application/pemkey-request Content Type Definition

(1) MIME type name: application

(2) MIME subtype name: pemkey-request

(3) Required parameters: none

(4) Optional parameters: none

(5) Encoding considerations: quoted-printable is always sufficient

(6) Security Considerations: none

The content of this body part corresponds to the following production.

```
    <request>       ::= <version>
                        ( <subject> / <issuer> / <certification> )
```

```
<version>       ::= "Version:" "5" CRLF
<subject>       ::= "Subject:" <id> CRLF
<issuer>        ::= "Issuer:" <id> CRLF
<certification> ::= "Certification:" <encbin> CRLF
```

This content type is used to provide for some of the request messages described in [6]. The information in the body part is entirely independent of any other body part. As such, the application/pemkey-request content type is an independent body part.

The certification request, certificate-retrieval request and crl-retrieval request are provided for directly. If the content contains a Certification: field it requests certification of the self-signed certificate in the field value. If the content contains an Issuer: field it requests the Certificate Revocation List (CRL) chain beginning with the CRL of the issuer identified in the field value. If the content contains a Subject: field it requests either the public key of the subject or a certificate chain beginning with the subject identified in the field value, or both if both exist.

The Subject: and Issuer: fields each contain a value of type <id>, which is defined in Section 2.2.

The crl-storage request is provided for by the application/pemkey-data content type described in the Section 5.2.

In each case, the response is transmitted in an application/pemkey-data content type. When returning public keys, certificate chains, and certificate revocation list chains, if there exists more than one, several application/pemkey-data body parts are to be returned in the reply message, one for each.

5.2. application/pemkey-data Content Type Definition

The principal objective of this content type is to convey cryptographic keying material from a source to a destination. However, no explicit provision is made for determining the authenticity or accuracy of the

data being conveyed. In particular, when a public key and its identifier is conveyed, there is nothing to prevent the source or an interloper along the path from the source to the destination from substituting alternate values for either the public key or the identifier, thus setting up the destination such that when the data is used sensitive information may be intercepted and disclosed inappropriately.

It is incumbent upon a recipient to verify the authenticity and accuracy of the data received prior to its use. The problem is addressed by the use of certificates, since a certification hierarchy is a well-defined mechanism that conveniently supports the automatic verification of the data. Alternatively, the application/key-data body part could be digitally signed by the source. In this way, if the destination believes that a correct source's public key is available locally and if the destination believes the source would convey accurate data, then the key data received from the source can be believed.

> NOTE: Insofar as a certificate represents a mechanism by which a third party vouches for the binding between a name and a public key, the signing of an application/pemkey-data body part is a similar mechanism.

(1) MIME type name: application

(2) MIME subtype name: pemkey-data

(3) Required parameters: none

(4) Optional parameters: none

(5) Encoding considerations: quoted-printable is always sufficient.

(6) Security Considerations: none

The content of this body part corresponds to the following production.

```
<keydata>       ::= <version>
                    ( <publickeydata> / <certchain> / <crlchain> )
<version>       ::= "Version:" "5" CRLF
<publickeydata> ::= "Key:" "PK" "," <publickey> "," <id-subset> CRLF
<certchain>     ::= <cert> *( [ <crl> ] <cert> )
<crlchain>      ::= 1*( <crl> [ <cert> ] )
<cert>          ::= "Certificate:" <encbin> CRLF
<crl>           ::= "CRL:" <encbin> CRLF
```

This content type is used to transfer public keys, certificate chains, or Certificate Revocation List (CRL) chains. The information in the body part is entirely independent of any other body part. (Note that the converse is not true: the validity of a protected body part cannot be determined without the proper public keys, certificates, or current CRL information.) As such, the application/pemkey-data content type is an independent body part.

The <publickeydata> production contains exactly one public key. It is used to bind a public key with its corresponding name form and key selector. It is recommended that when responders are returning this information that the enclosing body part be digitally signed by the responder in order to protect the information. The <id-subset> token is defined in Section 2.2.4.

The <certchain> production contains one certificate chain. A certificate chain starts with a certificate and continues with the certificates of subsequent issuers. Each issuer certificate included must have issued the preceding certificate. For each issuer, a CRL may be supplied. A CRL in the chain belongs to the immediately following issuer. Therefore, it potentially contains the immediately preceding certificate.

The <crlchain> production contains one certificate revocation list chain. The CRLs in the chain begin with the requested CRL and continue with the CRLs of subsequent issuers. The issuer of each CRL is presumed to have issued a certificate for the issuer of the preceding CRL. For each CRL, the issuer's certificate may be supplied. A certificate in the chain must belong to the issuer of the immediately preceding CRL.

The relationship between a certificate and an immediately preceding CRL is the same in both <certchain> and <crlchain>. In a <certchain> the CRLs are optional. In a <crlchain> the certificates are optional.

6. Examples

Given the following email message prepared for submission:

```
To: Ned Freed <ned@innosoft.com>
Subject: Hi Ned!

How do you like the new MIME/PEM?

Jim
```

When the text of the message is signed, it will look like this (note the use of the public key identifier with the included email name identifier):

```
To: Ned Freed <ned@innosoft.com>
Subject: Hi Ned!
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pem-signature";
        micalg="rsa-md5"; boundary="Signed Boundary"

--Signed Boundary
Content-Type: text/plain; charset="us-ascii"
Content-ID: <21436.785186814.2@tis.com>

How do you like the new MIME/PEM?

Jim

--Signed Boundary
Content-Type: application/pem-signature
Content-ID: <21436.785186814.1@tis.com>
Content-Transfer-Encoding: quoted-printable

Version: 5
Originator-ID: PK,MHkwCgYEVQgBAQICAwADawAwaAJhAMAHQ45ywA357G4fqQ61aoC1fO6B=
ekJmG4475mJkwGIUxvDkwuxe/EFdPkXDGBxzdGrW1iuh5K8kl8KRGJ9wh1HU4TrghGdhn0Lw8g=
G67Dmb5cBhY9DGwq0CDnrpKZV3cQIDAQAB,EN,2,galvin@tis.com
MIC-Info: RSA-MD5,RSA,PnEvyFV3sSyTSiGh/HFgWUIFa22jbHoTrFIMVERfMZXUKzFsHbmK=
tIowJlJR56OoImo+t7WjRfzpMH7MOKgPgzRnTwk0T5dOcP/lfbsOVJjleV7vTe9yoNp2P8mi/h=
s7
```

```
    --Signed Boundary--
```


If, instead, we choose to protect the headers with the text, it will look like this:

```
    To: Ned Freed <ned@innosoft.com>
    Subject: Hi Ned!
    MIME-Version: 1.0
    Content-Type: multipart/signed; protocol="application/pem-signature";
            micalg="rsa-md5"; boundary="Signed Boundary"

    --Signed Boundary
    Content-Type: message/rfc822
    Content-ID: <21468.785187044.2@tis.com>

    To:         Ned Freed <ned@innosoft.com>
    Subject:    Hi Ned!


    How do you like the new MIME/PEM?

    Jim

    --Signed Boundary
    Content-Type: application/pem-signature
    Content-ID: <21468.785187044.1@tis.com>
    Content-Transfer-Encoding: quoted-printable

    Version: 5
    Originator-ID: PK,MHkwCgYEVQgBAQICAwADawAwaAJhAMAHQ45ywA357G4fqQ61aoC1fO6B=
    ekJmG4475mJkwGIUxvDkwuxe/EFdPkXDGBxzdGrW1iuh5K8kl8KRGJ9wh1HU4TrghGdhn0Lw8g=
    G67Dmb5cBhY9DGwq0CDnrpKZV3cQIDAQAB,EN,2,galvin@tis.com
    MIC-Info: RSA-MD5,RSA,ctbDBgkYtFW1sisb5w4/Y/p94LftgQ0IrEn3d6WTwjfxFBvAceVW=
    fawsZPLijVKZUYtbIqJmjKtzTJlagBawfA/KhUsvTZdR6Dj+4Gd8dBBwMKvqMKTHAUxGXYxwNd=
    bK

    --Signed Boundary--
```

If we choose to encrypt the text of the following message, that is, encrypt the lines marked with asterick (*):


```
    To: Jim Galvin <galvin@tis.com>
    Subject: an encrypted message

    * How do you like the new MIME/PEM?
    *
    * Jim
```

the message would look as follows (note the use of the email name identifier):

```
    To: Jim Galvin <galvin@tis.com>
    Subject: an encrypted message
    MIME-Version: 1.0
    Content-Type: multipart/encrypted; protocol="application/pem-keys";
            boundary="Encrypted Boundary"

    --Encrypted Boundary
    Content-Type: application/pem-keys
    Content-ID: <21535.785187667.1@tis.com>
    Content-Transfer-Encoding: quoted-printable

    Version: 5
    DEK-Info: DES-CBC,D488AAAE271C8159
    Recipient-ID: EN,2,galvin@tis.com
    Key-Info: RSA,ISbC3IR01BrYq2rp493X+Dt7WrVq3V3/U/YXbxOTY5cmiy1/7NvSqqXSK/WZ=
    q05lN99RDUQhdNxXI64ePAbFWQ6RGoiCrRs+Dc95oQh7EFEPoT9P6jyzcV1NzZVwfp+u

    --Encrypted Boundary
    Content-Type: application/octet-stream
    Content-Transfer-Encoding: base64
```

```
    AfR1WSeyLhy5AtcX0ktUVlbFC1vvcoCjYWy/yYjVj48eqzUVvGTGMsV6MdlynUd4jcJgRnQIQvI
    m2VRgH8W8MkAlul+RWGu7jnxjp0sNsU562+RZr0f4F3K3n4wonUUP265UvvMj23RSTguZ/nl/Ox
    FM6SzDgV39V/i/RofqI=

    --Encrypted Boundary--
```

If, instead, we choose to sign the text before we encrypt it, the structure would be as follows (where lines with an asterick '*' are digitally signed and lines with an ampersand '&' are encrypted):


```
    Content-Type: multipart/encrypted; boundary="Encrypted Boundary"

    --Encrypted Boundary
    Content-Type: application/pem-keys

    KEY INFORMATION

    --Encrypted Boundary
    Content-Type: application/octet-stream

    & Content-Type: multipart/signed; boundary="Signed Boundary"
    &
    & --Signed Boundary
    & * Content-Type: text/plain
    & *
    & * How do you like the new MIME/PEM?
    & *
    & * Jim
    &
    & --Signed Boundary
    & Content-Type: application/pem-signature
    &
    & SIGNATURE INFORMATION
    &
    & --Signed Boundary--
```

```
    --Encrypted Boundary--
```

where KEY INFORMATION and SIGNATURE INFORMATION are descriptive of the actual content that would appear in a real body part. The actual message would be like this:


```
    To: Jim Galvin <galvin@tis.com>
    Subject: an encrypted message
    MIME-Version: 1.0
    Content-Type: multipart/encrypted; protocol="application/pem-keys";
            boundary="Encrypted Boundary"

    --Encrypted Boundary
    Content-Type: application/pem-keys
    Content-ID: <21546.785188458.1@tis.com>
    Content-Transfer-Encoding: quoted-printable

    Version: 5
    DEK-Info: DES-CBC,11CC89F8D90F1DFE
    Recipient-ID: EN,2,galvin@tis.com
    Key-Info: RSA,AZTtlEc6xm0vjkvtVUITUh7sz+nOuOwP0tsym6CQozD9IwVIJzY8+vIfbh5B=
    pR0kS6prq3EGFBFR8gRMUvbgHtEKPD/4ICQ7b6ssZ7FmKhl/cJC5rVjpb4EOUlwOXwRZ

    --Encrypted Boundary
    Content-Type: application/octet-stream
```

```
    Content-Transfer-Encoding: base64

    ZvWvtosDzRBXJzkDFFRb9Qjrgm2nDWg3zotJ3ZpExpWUG/aRJ7Vwd+PWkSfrDPJ52V/wkxwMrum
    xJHZonrtyd0AvaztvriMm2zXTefzwpGG1i5zK47PBqreLA3HDTK2U6B13vzpE8wMSVefzaCTSpX
    SCh08ceVEZrIYS53/CKZV2/Sga71pGNlux8MsJpYLwdj5Q3NKocg1LMngMo8yrMAe+avMjfOnhu
    49Xon1Gft+N5XDH/+wI9qxI9fkQvNZVDlWIhCYEkxd5ke549tLkJjEqHQbgJW5C+K/uxdiD2dBt
    nRCXcuO0Px3yKRyYg/9BgTf36padSHuv48xBg5YaqaEWpEzLI0Qd31vAyP23rqiPhfBn6sjhQ2K
    WhiF2l3TV8kQsIGHHZUkaUbqkXJe6PEdWWhwsqCFPDdkpjzQRrTuJH6xleNUFg+CG1V+tL4IgMj
    qm3KVojRXx8bG2auVN89NfwFswmoq4fXTrh3xyVS1VgxjKkcYI8SVVmkYjCxVviJP3zO2UzBvCo
    fADtBVBz1njYETtVGDO97uT39MqL85uEgiF4E5TkOj/m04+88G0/vvN/RISKJiFQJ3FyVIB/ShX
    Dixl8WCx3rxwN5g2QFLiyQVulzuNhimSD4ZxEo7smcTsAXUjwSLRtdjmTTutw2GmFESUaIrY81N
    pQJRPNAvF0IkN6ddwL4qvzUS99vjQp15g9FUv82lHtHwhM18a9GokVG8xYOjBBsn9anp9abh4Tp
    c/vpbunQUqnpV29rF4wj+8OwUOMi9ymGabBXAjw7DhNH2RdRVr1upQO896OX81VWB0LsA0cp+ym
    hTrEI+wCHcrsNMoRK/7zAeuAi0f1t9bN594EFlLoIrBnKEa1/OUAhMT7kG1fNkSRnc8BZswIoPy
    etsTurQfD40nsVHvNwE9Jz7wbBo00gd6blPADOUYFxfW5zu6ubygBqJiKPM4II2fCdNj7CptfQc
    RTeguKMVPLVmFg/EINuWBFm10GqlYT7p4zhfzysV/3r5LVZ1E8armTCRJ2GoYG5h+SKcytaQ0IT
    S2nLPCZl1hzdajsrqHFe8omQ

    --Encrypted Boundary--
```

In addition to text, the PEM services as defined here will protect arbitrary body parts, for example, the following audio body part:


```
    Content-Type: audio/basic

    AUDIO DATA HERE
```

when signed would appear as follows:

```
    Content-Type: multipart/signed; protocol="application/pem-signature";
            micalg="rsa-md5"; boundary="Signed Boundary"

    --Signed Boundary
    Content-Type: audio/basic
    Content-Transfer-Encoding: base64
```

```
    base64(AUDIO DATA HERE)

    --Signed Boundary
    Content-Type: application/pem-signature

    SIGNATURE INFORMATION HERE
```

and when encrypted would appear as follows:

```
    Content-Type: multipart/encrypted; protocol="application/pem-keys";
            boundary="Encrypted Boundary"

    --Encrypted Boundary
    Content-Type: application/pem-keys

    KEY INFORMATION HERE

    --Encrypted Boundary
    Content-Type: application/octet-stream
    Content-Transfer-Encoding: base64

    base64(encrypted(AUDIO DATA HERE))

    --Encrypted Boundary--
```

7. Observations

The use of the pre-submission and post-delivery processing steps to combine PEM and MIME capabilities exhibits several properties:

(1) It allows privacy-enhancement of an arbitrary content, not just the body of an RFC822 message.

(2) For a multipart or message content, it allows the user to specify different privacy enhancements to be applied to different components of the structure of the content.

(3) It provides for messages containing several privacy enhanced contents, thereby removing the requirement for PEM software to be able to generate or interpret a single content which intermixes both unenhanced and enhanced components.

The use of a MIME-capable user agent makes complex nesting of enhanced message body parts much easier. For example, the user can separately sign and encrypt a message. This motivates a complete separation of the confidentiality security service from the digital signature security service. That is, different key pairs could be used for the different services and could be protected separately.

This is useful for at least two reasons. First, some public key algorithms do not support both digital signatures and encryption, for example, the way that the RSA algorithm does; two key pairs would be required in this case. Second, an employee's company could be given access to the (private) decryption key but not the (private) signature key, thereby granting the company the ability to decrypt messages addressed to the employee in emergencies without also granting the company the ability to sign messages as the employee.

8. Summary of Changes to PEM Specification

This document updates the message encryption and signature procedures defined by [3] and replaces the key certification and related services defined by [6]. The changes are enumerated below.

(1) The PEM specification currently requires that encryption services be applied only to message bodies that have been signed. By providing for each of the services separately, they may be applied recursively in any order according to the needs of the requesting application.

(2) PEM implementations are currently restricted to processing only text-based electronic mail messages. In fact, the message text is required to be represented by the ASCII character set with "<CR><LF>" line delimiters. This restriction no longer applies.

(3) MIME includes transfer encoding operations to ensure the unmodified

transfer of body parts, which obviates these services in PEM.

(4) PEM specifies a Proc-Type: header field to identify the type of processing that was performed on the message. This functionality is subsumed by the MIME Content-Type: headers. The Proc-Type: header also included a decimal number that was used to distinguish among incompatible encapsulated header field interpretations which may arise as changes are made to the PEM standard. This functionality is replaced by the Version: header specified in this document.

(5) PEM specifies a Content-Domain: header, the purpose of which is to describe the type of the content which is represented within a PEM message's encapsulated text. This functionality is subsumed by the MIME Content-Type: headers.

(6) The PEM specifications include a document that defines new types of PEM messages, specified by unique values used in the Proc-Type: header, to be used to request certificate and certificate revocation list information. This functionality is subsumed by two new content types specified in this document.

(7) The header fields having to do with certificates (Originator-Certificate: and Issuer-Certificate:) and CRLs (CRL:) are relegated for use only in the application/pemkey-data and application/pemkey-request content types and are no longer allowed in the header portion of a PEM signed or encrypted message.

(8) The grammar specified here explicitly separates the header fields that may appear for the encryption and signature security services. It is the intent of this document to specify a precise expression of the allowed header fields; there is no intent to disallow the functionality of combinations of encryption and signature security found in [3].

(9) With the separation of the encryption and signature security services, there is no need for a MIC-Info: field in the headers associated with an encrypted message.

(10) In [3], when asymmetric key management is used, an Originator-ID field is required in order to identify the private key used to sign the MIC argument in the MIC-Info: field. Because no MIC-Info: field is associated with the encryption security service under asymmetric key managment, there is no requirement in that case to

include an Originator-ID field.

These changes represent a departure in mechanism, not in effect, from those specified in [3] and [6]. The following technical changes to [3] and [4] are also specified by this document.

(1) The grammar specified here explicitly excludes symmetric key management. Currently, there are no generally available implementations of symmetric key management nor are there any known plans for implementing it. As a result, the IETF standards process will require this feature to be dropped when the documents are promoted to draft standard status from proposed standard status.

(2) This document requires all data that is to be digitally signed to be represented in 7bit form.

(3) This document broadens the allowable name forms that users may use to identify their public keys. Users may use arbitrary strings and email addresses as their name. Further, users may distribute their public key directly in lieu of using certificates. In support of this change the Originator-ID-ASymmetric: and Recipient-ID-ASymmetric: fields are deprecated in favor of Originator-ID: and Recipient-ID: fields, respectively.

9. Collected Grammar

The version of the grammar in this document is as follows:

```
<version>       ::= "Version:" "5" CRLF
```

The content of an application/pem-signature body part is as follows:

```
<pemsig>        ::= <version> ( 1*<origasymflds> )

<origasymflds>  ::= <origid> <micinfo>

<origid>        ::= "Originator-ID:" <id> CRLF
```

The content of an application/pem-keys body part is as follows:

```
    <pemkeys>       ::= <version> <dekinfo> 1*<recipasymflds>

    <recipasymflds> ::= <recipid> <asymkeyinfo>

    <recipid>       ::= "Recipient-ID:" <id> CRLF

    <asymkeyinfo>   ::= "Key-Info" ":" <ikalgid> "," <asymencdek> CRLF
```

Identifiers are defined as follows:

```
    <id>            ::= <id-subset> / <id-publickey> / <id-issuer>

    <id-subset>     ::= <id-email> / <id-string> / <id-dname>

    <id-email>      ::= "EN"  "," <keysel> "," <emailstr> CRLF

    <id-string>     ::= "STR" "," <keysel> "," <string> CRLF

    <id-dname>      ::= "DN"  "," <keysel> "," <dnamestr> CRLF

    <id-publickey>  ::= "PK"  "," <publickey> [ "," <id-subset> ] CRLF

    <id-issuer>     ::= "IS"  "," <dnamestr>  "," <serial> CRLF

    <keysel>        ::= <encbin>
                        ; a printably encoded non-null sequence of octets

    <emailstr>      ::= <addr-spec> / <route-addr>
                        ; an electronic mail address as defined by
                        ; these two tokens from RFC822

    <string>        ::= ; a non-null sequence of characters

    <dnamestr>      ::= <encbin>
                        ; a printably encoded, ASN.1 encoded
                        ; distinguished name

    <publickey>     ::= <encbin>
                        ; a printably encoded, ASN.1 encoded
                        ; subjectPublicKeyInfo
```

```
    <serial>        ::= 1*<hexchar>
                        ; hex dump of the serial number of a certificate
```


The content of an application/pemkey-request body part is as follows:

```
    <request>       ::= <version>
                        ( <subject> / <issuer> / <certification> )

    <subject>       ::= "Subject:" <id> CRLF

    <issuer>        ::= "Issuer:" <id> CRLF

    <certification> ::= "Certification:" <encbin> CRLF
```

The content of an application/pemkey-data body part is as follows:

```
    <keydata>       ::= <version>
                        ( <publickeydata> / <certchain> / <crlchain> )

    <publickeydata> ::= "Key:" "PK" "," <publickey> "," <id-subset> CRLF

    <certchain>     ::= <cert> *( [ <crl> ] <cert> )

    <crlchain>      ::= 1*( <crl> [ <cert> ] )

    <cert>          ::= "Certificate:" <encbin> CRLF

    <crl>           ::= "CRL:" <encbin> CRLF
```

10. Security Considerations

This entire document is about security.

11. Acknowledgements

David H. Crocker suggested the use of a multipart structure for MIME-PEM

interaction.

12. References

[1] David H. Crocker. Standard for the Format of ARPA Internet Text Messages. RFC 822, University of Delaware, August 1982.

[2] Nathaniel Borenstein and Ned Freed. MIME (Multipurpose Internet Mail Extension) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC 1521, Bellcore and

Innosoft, September 1993. Obsoletes RFC 1341.

[3] John Linn. Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. RFC 1421, February 1993. Obsoletes RFC 1113.

[4] Steve Kent. Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. RFC 1422, BBN Communications, February 1993.

[5] David M. Balenson. Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers. RFC 1423, Trusted Information Systems, February 1993.

[6] Burton S. Kaliski. Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services. RFC 1424, RSA Laboratories, February 1993.

[7] James Galvin, Sandy Murphy, Steve Crocker, and Ned Freed. Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. RFC XXXX, Trusted Information Systems and Innosoft, XXXX 1994.

13. Authors' Addresses

Steve Crocker
email: crocker@tis.com

James M. Galvin
email: galvin@tis.com

Sandra Murphy
email: murphy@tis.com

Trusted Information Systems
3060 Washington Road
Glenwood, MD 21738
Tel: +1 301 854 6889
FAX: +1 301 854 5363

Ned Freed
Innosoft International, Inc.
1050 East Garvey Avenue South
West Covina, CA 91790
Tel: +1 818 919 3600
FAX: +1 818 919 3614
email: ned@innosoft.com

14. Appendix: Imported Grammar

The following productions are taken from [3]. The grammar presented in [3] remains the authoritative source for these productions; they are repeated here for the convenience of the reader.

```
    <dekinfo>    ::= "DEK-Info" ":" <dekalgid> [ "," <dekparameters> ] CRLF

    <micinfo>    ::= "MIC-Info" ":" <micalgid> "," <ikalgid> ","
                     <asymsignmic> CRLF

    <encbin>     ::= 1*<encbingrp>
    <encbingrp>  ::= 4*4<encbinchar>
    <encbinchar> ::= ALPHA / DIGIT / "+" / "/" / "="
```

The following productions are taken from [5]. The grammar presented in

[5] remains the authoritative source for these productions; they are repeated here for the convenience of the reader.

```
    <dekalgid>          ::= "DES-CBC"
    <ikalgid>           ::= "DES-EDE" / "DES-ECB" / "RSA"
    <micalgid>          ::= "RSA-MD2" / "RSA-MD5"

    <dekparameters>     ::= <DESCBCparameters>
    <DESCBCparameters>  ::= <IV>
    <IV>                ::= <hexchar16>

    <asymsignmic>       ::= <RSAsignmic>
    <RSAsignmic>        ::= <encbin>

    <asymencdek>        ::= <RSAencdek>
    <RSAencdek>         ::= <encbin>

    <hexchar16>         ::= 16*16<hexchar>
    <hexchar>           ::= DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
                                                    ; no lower case
```

The following productions are taken from [1]. The grammar presented in [1] remains the authorative source for these productions; they are repeated here for the convenience of the reader.


```
    <addr-spec>     ::= <local-part> "@" <domain>        ; global address

    <local-part>    ::= <word> *( "." <word> )           ; uninterpreted
                                                         ; case-preserved

    <domain>        ::= <sub-domain> *( "." <sub-domain> )

    <sub-domain>    ::= <domain-ref> / <domain-literal>

    <domain-ref>    ::= <atom>                           ; symbolic reference
```

```
    <route-addr>    ::= "<" [ <route> ] <addr-spec> ">"

    <route>         ::=  1# ( "@" <domain> ) ":"         ; path-relative


    <word>          ::= <atom> / <quoted-string>

    <quoted-string> ::= """ *( <qtext> / <quoted-pair> ) """

    <qtext>         ::= (any <CHAR> excepting """, "
                         and including <linear-white-space>)

    <quoted-pair>   ::= "

    <linear-white-space> ::= 1*( [ CRLF ] <LWSP-char> )  ; semantics = SPACE
                                                         ; CRLF => folding
    <LWSP-char>     ::= SPACE / HTAB                     ; semantics = SPACE


    <atom>          ::= 1*(any <CHAR> except <specials>, SPACE and <CTL>s)

    <CHAR>          ::= <any ASCII character>

    <CTL>           ::= <any ASCII control character and DEL>

    <specials>      ::=  "(" / ")" / "<" / ">" / "@"     ; Must be in quoted-
                         /  "," / ";" / ":" / "
                         /  "." / "[" / "]"              ;  within a word.
```


Table of Contents

Status of this Memo 1
Abstract 1

1 Introduction 2
2 Name Forms and Identifiers 4
2.1 Name Forms 4
2.1.1 Email Addresses 5
2.1.2 Arbitrary Strings 5
2.1.3 Distinguished Names 5
2.2 Identifiers 6
2.2.1 Email Address 7
2.2.2 Arbitrary String 7
2.2.3 Distinguished Name 7
2.2.4 Public Key 7
2.2.5 Issuer Name and Serial Number 8
3 Applying PEM Security Services to MIME Body Parts 8
3.1 PEM Processing Steps 9
3.2 Use of multipart/signed Content Type 11
3.3 Use of multipart/encrypted Content Type 12
3.4 application/pem-signature Content Type Definition 12
3.5 application/pem-keys Content Type Definition 13
4 Removing PEM Security Services from PEM Body Parts 14
5 Key Management Content Types 15
5.1 application/pemkey-request Content Type Definition 15
5.2 application/pemkey-data Content Type Definition 17
6 Examples 19
7 Observations 24
8 Summary of Changes to PEM Specification 25
9 Collected Grammar 27
10 Security Considerations 29
11 Acknowledgements 29
12 References 29
13 Authors' Addresses 30
14 Appendix: Imported Grammar 32