An Integration of PGP and MIME

Kazuhiko YAMAMOTO
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma City 630-01 JAPAN
Email: kazu@is.aist-nara.ac.jp

Abstract

Internet text mail has been developing to satisfy various user requests, such as transporting non-textual objects and privacy enhancements. While MIME redefined the mail body format to support non-textual objects and multipart structure, PGP provides encryption and digital signature features for text mail. MIME, however, does not provide privacy services whereas non-textual objects cannot be exchanged with PGP. It is of recent interest to integrate PGP and MIME so that users can make use of these two services at the same time. This paper describes an integration of PGP and MIME. Our scheme embeds PGP objects into MIME and maintains backward compatibility with PGP. It is possible to encrypt, sign, and sign-then-encrypt non-textual objects, singleparts in a multipart, an entire multipart, etc. We also explain our viewing and composing mechanisms that allow users to handle PGP/MIME messages intuitively without format restrictions.

1 Introduction

Since RFC822 [1] specified the format of Internet text mail in 1982, it has widely spread and is now a representative of Internet services. Due to poor computing power and unstable network systems at that time, RFC822 messages limited their mail body to text. As the computing power and capacities improved, the user's desire to transport non-textual objects such as pictures, video, and audio increased. Many people from various countries extended RFC822 messages to contain non-English characters from their native language. To satisfy the diversity of user requests and to bridge the localized RFC822 messages, Multipurpose Internet Mail Extensions(MIME) was proposed in 1992 [2].

Since the Internet has been changing from research oriented to commercial and from organizational to individual, privacy for electronic mail has become a significant problem. Pretty Good Privacy(PGP) [3] was released in 1991 with the hope of providing confidentiality, message integrity, user authentication and non-repudiation for electronic mail. PGP is now supported not only by users in the US but in other countries as well.

It is natural that users would want to use the rich features of MIME and the privacy services of PGP at the same time. Unfortunately, MIME and PGP have taken separate evolution paths. MIME itself does not provide privacy enhancement services whereas non-textual objects cannot be exchanged with PGP. So, it is of recent interest to integrate PGP and MIME. In this paper, we propose an integration of PGP and MIME, embedding PGP objects within MIME. We call our integration scheme "PGP/MIME", while we use the term "PGP/RFC822" to indicate an RFC822 message containing one PGP object. It is possible to encrypt, sign, and sign-then-encrypt non-textual objects, singleparts in a multipart, an entire multipart, etc in our scheme.

IETF took another approach to enhance privacy for MIME by integrating Privacy Enhanced Mail(PEM) [4] and MIME. It was known as PEM/MIME but is now called MIME Object Security Services(MOSS) [5]. MOSS maps a target object and its control block into a MIME multipart. This format is elegant but results in sacrificing backward compatibility with PEM that targets RFC822 messages. This is why MOSS stopped using the word PEM. Since PGP is now the de facto standard, such a departure would be a tragedy for PGP. Our integration scheme does not require any modifications to PGP at all, so it maintains minimum compatibility with PGP/RFC822.

The key to spreading security services for MIME is not in the elegance of the internal syntax but of the user interface. It is crucial to provide an easy-to-use viewer and an intuitive composer to MIME users enhancing security features. Our PGP/MIME viewer provides a simple but powerful interface. It automatically decodes any complicated PGP/MIME message and recursively displays the analyzed syntax with PGP warnings. Users can read any part in any order and reply to an encrypted message as if it was plain text. Our PGP/MIME composer maps file structure to MIME format, processing PGP according to user specified marks. Though it does not requires users to understand a composition grammar, users can create any complicated PGP/MIME message without format limitations.

Throughout this paper, we use the acronyms "CT:", "CTE:", and "CD:" to express "Content-Type:", "Content-Transfer-Encoding:", and "Content-Description:" respectively. While "MIME encoding" indicates encoding mechanisms provided by MIME such as "base64" and "quoted-printable", "PGP encoding" means encrypting, signing, or signing-then-encrypting by PGP.

This paper is organized as follows: Section 2 specifies the format of PGP/MIME. We explain the implementation of our PGP/MIME viewer and composer in Section 3. Section 4 gives evaluations of our design and implementation of PGP/MIME. We describe implementation status in Section 5 and conclude this paper in Section 6.

2 Format of PGP/MIME

This section describes the format of PGP/MIME. It is designed to protect MIME objects with PGP and to enclose PGP objects within the context of MIME messages. For backward compatibility, our scheme makes use of one Content-Type:, "application/pgp". As far as the author knows, this kind of approach was originally found in the withdrawn Internet-Draft entitled "An Alternative PEM MIME Integration" by Schiller in 1993, though it did not specify the "format" parameter. Our PGP/MIME is based on the withdrawn Internet-Draft entitled "The application/pgp MIME Content-type" by Borenstein et al in 1994, but encoding mechanisms are different. This section describes the syntax of a MIME object whose content body is a PGP object, which includes a text or a MIME object.

2.1 Definition of Application/PGP

PGP/MIME uses "CT: application/pgp" to enclose a PGP object within MIME. The CT: may take one parameter "format", whose value is "text" or "mime". This parameter makes it possible to embed not only text but also a MIME object in the PGP object. If the value of format parameter is "text", it indicates that the PGP object contains text. Otherwise a PGP object holds a MIME object. If the parameter is omitted, it is identical to "format=text".

A PGP/RFC822 message can be converted to a PGP/MIME message with "Mime-Version: 1.0', "CT: application/pgp", and optional "CTE:" fields. We call this kind of PGP/MIME messages *conventional* PGP/MIME. Note that user IDs to select public keys for encryption are equal to mail addresses on a mail header such as To: or Cc: for a PGP/RFC822 message or a conventional PGP/MIME message because it has only one PGP object.

To maintain minimum backward compatibility, if the parameter is omitted or is "text"(i.e. a conventional PGP/MIME message), the PGP object is assumed to contain *localized* text, which is not always US-ASCII. It can be ISO-8859-1, ISO-2022-JP or whatever. The decoded text by PGP should be treated according to local convention. For example, we assumed that the text encoded by PGP is ISO-2022-JP in Japan.

If a target object is not text, it should be converted into a MIME object which consists of a content header and a content body. Then, the MIME object is encoded by PGP to get a PGP object, which is included by a MIME object whose CT: is application/pgp.

Note that we cannot decide user IDs for public keys from the mail header in PGP/MIME. For example, one part can be encrypted for one person and another part can be encrypted for another person while the whole message is destined to a mailing list.

2.2 CTE: considerations

PGP provides radix64 encoding, which is syntactically identical to MIME base64 encoding but flagged in different manners. The PGP mechanism is self-identifying, while the MIME mechanism uses CTE: to indicate an encoding type. There are two methods to convert a target object to a "mail-safe" form. One is to encode a PGP output by MIME encoding. The other is that PGP itself converts an object to a mail-safe form and CTE: just indicates an encoding domain(i.e. 7bit or 8bit). Note that "7bit" and "8bit" specified in CTE: means that no encoding is applied to its object.

Since the content header (whose content body is a PGP object) cannot be protected with our scheme, it is quite possible for someone to forge or modify CTE: in the content header. So, it is safer for MIME readers to pass the PGP object in the content body to a PGP process without MIME decoding. For this reason, we make use of PGP's mail-safe features rather than MIME encoding. (Note that CT: in the content header is also under threat of modification. Unfortunately, our scheme is vulnerable to this kind of attack.)

PGP objects are categorized into three types. Signed objects by PGP are 7bit or 8bit. For example, 7bit text such as ISO-2022-JP results in a clear signature in 7bit whereas 8bit text such as ISO-8859-1 are tranformed into a clear signature in 8bit. A binary object is converted into 7bit text with radix64 encoding after the calculation of its signature.

Note that an object must be encoded into 7bit representation(i.e. 7bit, quoted-printable, or base64) before the signature calculation in the MOSS scheme. But such preprocessing is optional in PGP/MIME. If 7bit transformation is always required, a clear signature cannot be created from 8bit text. This limitation is very inconvenient to those who usually use 8bit text to express their native language. 8bit clear signatures have been used for a long time, so we cannot sacrifice backward compatibility by forcing 7bit representation. Usually a signature for ISO-8859-1 text is created without the format parameter. If the transport system does not support 8bit text, the localized 8bit text should be converted into a MIME object with a proper CTE: before it is passed to PGP.

The domain of encrypted and signed-then-encrypted objects by PGP is always 7bit with radix64 encoding. The MIME object whose content body is a PGP object should provide CTE: according to the encoding domain of the PGP object. CTE: 7bit can be omitted but CTE: 8bit must be provided.

2.3 Canonicalization

Since each operating system has their own type of line break, line breaks of a target object must be canonicalized before PGP calculates a digital signature and/or encrypts a target object for interoperability. When the -t option is specified, PGP first canonicalizes each line break to CRLF. This line break is identical to that of RFC822, so we can make use of this PGP feature. If a target object is localized text, we should, of course, execute PGP with the -t option.

If a target object is not text, we first convert it to a MIME object preparing an appropriate content header. It is possible to convert 8bit text to a MIME object for transport system-safe. If the MIME object is "text", "multipart", "application/postscript", or "message", it must be passed to PGP as a line-based object. So, if the original object is in the binary domain, it must be encoded to the 7bit domain when it is tranformed to a MIME object.

Since multipart and message types allow recursive structure, MIME prohibits encoding of an entire object. So the CTE: must be 7bit, 8bit or binary. In order to pass multipart and message to PGP as a line-based object, they must not include objects in the binary domain. Objects in the binary domain must be encoded by MIME before they are enclosed in a multipart or message.

Other MIME objects in 7bit or 8bit domain should be treated as line-based objects by PGP. If CTE: is "binary", it must be passed to PGP as binary. It is not necessary to apply

MIME encoding to the original object in the binary domain before it is encoded by PGP. Since we do not specify the -t option for MIME objects in the binary domain, line breaks of the content header must be converted to CRLF before this object is passed to PGP.

3 Implementation of PGP/MIME

It is crucial to provide an easy-to-use viewer and an intuitive composer to MIME users privacy functionality so that security services in MIME will be widely used. This section describes a novel PGP/MIME interface, "Mew" (Message interface to Emacs Window), which works on Emacs. We first explain Mew's PGP/MIME composer in Section 3.1, then describe a viewer in Section 3.2. Since all methods in this section are independent on the spec of PGP/MIME, they are applicable to other privacy enhanced MIME schemes such as MOSS.

3.1 Composer

Many MIME composers define their own complicated composition grammar or force complex command line options to compose MIME messages. Complicated operations are not only hard to use but are also prone to miss operations. Moreover, most composers fail to provide methods to support deep multipart. Such a complicated and imperfect composing system will confuse users especially when composing PGP/MIME messages. Thus, requirements for PGP/MIME composers can be summarized as follows:

- The PGP/MIME composer must be able to compose PGP/MIME messages with easy operations.
- The PGP/MIME composer must not define a complicated composition grammar that is hard to understand.
- The PGP/MIME composer must not require the users to understand MIME or other syntax.
- The PGP/MIME composer must be able to compose PGP/MIME syntax without any limitations.

Mew provides two methods for composing a PGP/MIME message. One is a mark based method for creating any kind of PGP/MIME message. The other is a shortcut to handle only localized text. We first describe the shortcut method, then explain the mark based composing.

3.1.1 A shortcut for conventional PGP/MIME

Since users mostly use localized text in daily life, localized text with PGP protection has been exchanged. So, it is a good idea to create the most used PGP/MIME messages without any troublesome operations. Non-MIME viewers treat conventional PGP/MIME exactly as PGP/RFC822. Note that we do not require pre-encoding to 8bit text, so conventional PGP/MIME messages are completely backward compatible with non-MIME viewers. Note also that MIME viewers which do not support PGP/MIME treat the messages as text/plain.

Mew provides three commands to create a conventional PGP/MIME message. Each cuts the mail body in a draft buffer to pass it to PGP, then insert the PGP output to the draft buffer in turn. The signature function asks a user to input his passphrase. This passphrase is never echoed back and is delivered to PGP interactively. It should be noted that if a passphrase is given to PGP as a command line argument or via an environment variable, the passphrase may be monitored by local eavesdroppers on a multi-user OS. Thus, the passphrase must be sent to PGP interactively to prevent eavesdropping by non-privileged users. But we should keep in mind that privileged users may still monitor a keyboard or pipe stream.

```
To: kazu@is.aist-nara.ac.jp
Subject: PGP signed message
Mime-Version: 1.0
----
This body is signed by PGP.

---
keiichi
```

Figure 1: An example draft

The function for encryption automatically extracts user IDs from To: and Cc: fields to specify receivers to PGP. Note that the sender's user ID is also specified so that the sender can decrypt the back up message. The function for sign-then-encrypt executes PGP with extracted user IDs then passes the input passphrase to PGP.

Since Mew runs on Emacs, each command is bound to a key. Figure 1 shows an example of an RFC822 message and Figure 2 illustrates a PGP/MIME message after a signature function is executed and a passphrase is input in the mini buffer.

3.1.2 Mark based composing

Mew's MIME composer provides a simple yet powerful composing of file structure mapping to MIME syntax. That is, directories correspond to multipart and files indicate single part. A user can create any complex multipart with file operations such as copy, link, remove, and make a directory, which are bound to single keys. The default CT: is determined by the suffix of the filename. For example, application/postscript is selected for the file "cat.ps". Encoding strategy is decided by pre-defined rules. For example, base64 is chosen for audio/basic, quoted-printable for ISO-8859-1 text. The user can change CT:, CD:, and CTE: at any time. Multipart/Mixed is chosen for the default CT: of directories. Figure 3 is an example of a draft buffer of a multipart, depth 2.

In addition to the mail header and the mail body, multipart structure is displayed at the bottom of the draft buffer. This region is prepared according to the user's instruction. Key bindings of the region are different from that of the mail header and the mail body. The first column consists of marks that indicate encoding(e.g. "B" for base64 and "Q" for quoted-printable). The next column indicates the part number where numbers for directories always end with "0". The third column shows file or directory names. Note that the directory on the first line of the region indicates the entire multipart message, which must be readable only by

```
To: kazu@is.aist-nara.ac.jp
Subject: PGP signed message
Mime-Version: 1.0
Content-Type: application/pgp
----
-----BEGIN PGP SIGNED MESSAGE-----

This body is signed by PGP.

- ---
keiichi

-----BEGIN PGP SIGNATURE-----
Version: 2.6.i

iQCVAgUBMCtrNhTyAnmgatc9AQFK5gP/Zyptfl1cX+OkbULgrUNkuOAhL4Wok+vJ
OPrD3TSIFZ/lh3T/Hjtjq6I6PELDKI9CXJFZRKgyCZhBCZRdXJP5yaWuC5S4gJNu
+zlLqs2TupfWJrK+wndRKP5N2DyxnxX3dd5CZhu9C1220/lV18zvIl5Vie0cowAe
1y/NyfxiuUs=
=hZka
-----END PGP SIGNATURE-----
```

Figure 2: A conventional PGP/MIME message

```
To: kazu@is.aist-nara.ac.jp
Subject: Cats
Mime-Version: 1.0
----
This is my cat.

---- multipart --
    0   1/               Multipart/Mixed
    1     00CoverPage      Text/Plain
    2.0   dir/             Multipart/Mixed
B   2.1     cat.gif          image/gif            "A pretty cat"
Q   2.2     cat.ps           application/postsc..
---- multipart ----
```

Figure 3: Composing a complicated MIME message

the user for security reasons. The fourth column shows CT: and the last column indicates CD:.

Just before sending the message, a MIME message is automatically created according to the user specified files, each CT:, each CD:, and each mark. In Emacs, ISO-8859-1 is automatically chosen as the "charset" parameter for 8bit text, otherwise US-ASCII is selected. In Mule(MULtilingual Enhancement to GNU Emacs)[6], the charset is guessed from the Mule internal multi-lingual representation.

Mew's composer integrates MIME encoding and PGP encoding, which are displayed as marks. In addition to the marks "B" and "Q", the marks "PE", "PS", and "PSE", which indicated PGP encrypt, sign, and sign-then-encrypt respectively, are provided. Note that a PGP mark on directory means that PGP encoding is applied to the entire multipart. When the user puts "PE" or "PSE" marks on any part, the user is asked to specify receivers. The user enters comma-separated user IDs in the mini buffer and then the information is displayed on the last column, overriding CD:. Note that the sender's user ID is additionally specified during encryption so that the sender can decrypt a backup message. Figure 4 illustrates an example of PGP/MIME composing. The "PE" mark is given to part 2.0 and "PS" is put on parts 1 and 2.1. Note that the user can cancel the marks at any time.

Mew maps the given file tree to PGP/MIME in postorder executing PGP as Emacs subprocesses corresponding to PGP marks. For example, the file tree in Figure 4 is converted to PGP/MIME format as follows: First the region of the mail body is stored as a file name of "00CoverPage" in the directory "1" to complete the file tree. Next Mew walks around the directory "1" in postorder to create a multipart. The charset is guessed for 00CoverPage since it is text, it is then signed by PGP. At this time, the user is required to input a passphrase. Next Mew goes down to the directory "dir" to create another multipart. When Mew passes "cat.gif" to PGP to calculate a signature, the user is asked to enter the passphrase again because the previous passphrase is not reused to prevent eavesdropping. After "cat.ps" is encoded quoted-printable, the second multipart is constructed. Mew then sends this multipart to PGP for encryption. After this step is completed, the outer multipart is prepared. Since the directory does not have a mark, the whole process is finished.

3.2 Viewer

Some MIME viewers provide full MIME functionality but many of them force users to read parts in the composed order. This frustrates users who want to read any part in any order. Since most viewers never cache analyzed MIME syntax, users also become frustrated if they have to read the same message repeatedly. This is very inconvenient, especially for PGP/MIME because users are always requested to input their passphrase every time they read encrypted PGP/MIME messages. It is natural that PGP/MIME users want to reply and cite PGP/MIME messages as if they were plain text. But PGP/MIME message should be stored in a disk storage with PGP protected format for security reasons. We thus summarize requirements for PGP/MIME viewers as follows:

- The PGP/MIME viewer must provide users with rich operations for each part.
- The PGP/MIME viewer must quickly display a PGP/MIME message if it is read repeatedly.
- The PGP/MIME viewer must be able to treat a PGP/MIME message as plain text but store the message in a PGP protected format in disk storage.

We first describes the internal mechanism of Mew's PGP/MIME viewer in Section 3.2.1 and then explain how to handle PGP waring in Section 3.2.2

3.2.1 Internals of Mew's PGP/MIME viewer

Mew's viewer consists of a local form decoder, a MIME syntax analyzer, and a displayer. When a user reads a MIME message, Mew copies the message into a cache buffer of Emacs. The local form decoder decodes MIME format according to CTE: to obtain raw data. If CT: is application/pgp, it decrypts or verifies PGP objects. Every time a PGP object is decrypted, the user is requested to input his passphrase, that is, the passphrase is not reused to prevent eavesdropping. The PGP decoded object is recursively decoded according to CTE: if the format is "mime". When Mew runs on Mule, the decoder transforms enveloped multi-lingual text to the Mule internal character representation according to the charset parameter. Mew also converts a conventional PGP/MIME message to Mule internal character representation following a local convention. In this way, the decoder decodes a PGP/MIME message in preoder so that each part has a native data image.

Next the analyzer analyzes the structure of the locally formatted message recursively. It saves the analyzed syntax as a local variable of the cache buffer. Cache buffers are managed as an LRU list and the list-size is customizable. If the message is a singlepart, the displayer displays the singlepart according to CT:. So, a conventional PGP/MIME message is simply displayed in a message buffer. If the message is multipart, the displayer simply shows the structure in a summary buffer so that the user can select any part. Figure 5 is an example of a displayer displaying a message syntax in the summary buffer corresponding to Figure 4.

The user can move the cursor onto any part he or she wishes and then display the part in any order. Since the message has been already decoded and stored in the cache buffer, the user can reply and cite the message as if it was plain text. The user is not required to input the passphrase the next time the message is read unless expires from the cache.

Mew does not automatically decode PGP/RFC822 by PGP because it does not have CT: application/pgp. Mew does provide a function to decode a message by PGP manually so that the user can decode PGP/RFC822 to obtain localized text.

3.2.2 Warning handling

One of most important functionalities of enhancing privacy services for MIME is to report the results of verification of a digital signature. PGP reports the success of verification as a "Good signature". If any alteration is found, a "Bad signature" warning is returned. PGP's key management system provides a grassroot web of trust. The highlight of this system is *validity* of a public key, an indication that the key

```
To: kazu@is.aist-nara.ac.jp
Subject: Cats
Mime-Version: 1.0
----
This is my cat.

---- multipart --
    0   1/               Multipart/Mixed
PS  1     00CoverPage      Text/Plain
PE  2.0   dir/             Multipart/Mixed        "kazu"
PS  2.1     cat.gif          image/gif            "A pretty cat"
Q   2.2     cat.ps           application/postsc..
---- multipart ----
```

Figure 4: Mark based composing for PGP/MIME

```
   1 M08/11 keiiti-s@is.aist- Cats <<----Next_Part(Fri_Aug_11_23:43:52_1995)--
        1       Text/Plain
        2.1     image/gif       "A pretty cat"
        2.2     application/postscript
```

Figure 5: PGP/MIME syntax displayed in the summary buffer

actually belongs to the person to whom it says it belongs. PGP warns the user if the validity of a public key is not complete. Mew is designed to report the value of validity — complete, marginal, untrusted, or undefined. Since Mew automatically decrypts encrypted messages by PGP, users may not notice they are encrypted. So, Mew notifies users of the parts of a PGP/MIME message that are encrypted by PGP.

An earlier version of Mew, that supported only conventional PGP/MIME, inserted the report of PGP into the bottom of the message. This approach is no longer practical for PGP/MIME because an object encoded by PGP is not restricted to text. A binary object is destroyed if the PGP report is inserted into the bottom of it.

So, Mew makes use of content headers of each part to hold the report of PGP. After the decoder decodes a PGP object, it inserts an "X-Mew:" field whose value indicates the report of PGP. The analyzer corrects X-Mew: fields analyzing MIME syntax and saves them as a part of MIME syntax to the local variable of the cache buffer. The displayer inserts corrected X-Mew: fields to the mail header when the mail header is displayed so that the user can see the PGP report first. This fields should not be stored statically since validity of public keys will change. A sly cracker could insert illegal X-Mew: fields to deceive the receivers. So, the decoder carefully removes X-Mew: fields first.

Figure 6 shows an example of a PGP warning corresponding to Figure 4. The number in angle brackets indicates the part number. If the number is omitted, the warning is for the entire message. The first line tells us that part 1 is signed by "Keiiti" whose public key's validity is complete and that the verification succeeded. The second line shows that part 2 was encrypted multipart. Part 2.1 contains a good signature by Keiiti. Figure 8 gives a snapshot image of the PGP/MIME viewer displaying the message of Figure 4. The GIF image of a cat, which is signed by Keiiti, is displayed by an image viewer.

If the public key of an originator is not found in the recipient's keyring or the keyring itself does not exist, signature verification fails. If a PGP/MIME message is not encrypted for the recipient, PGP cannot decrypt it. Mew reports such causes of PGP decoding failure to X-Mew: field.

4 Evaluations and experiences

The syntax of our PGP/MIME is highly dependent on the implementation of PGP. Since PGP can distinguish whether a PGP object is encrypted, signed, or signed-then-encrypted by itself, we do not prepare a parameter to identify the PGP services. Schiller's Internet draft prepared a new value for the PEM parameter "Content-Domain:" instead of the MIME parameter "format". This approach is not practical to PGP/MIME for two reasons. For backward compatibility with PGP, we must not modify the PGP program at all, so any new parameter or value should not be defined for PGP. The other reason is that such a PGP parameter is meaningless for actual implementation. Consider a MIME object containing a PGP object, which embeds a MIME object. The decoding process is as follows: After a MIME viewer removes the outer content header, it passes the content body to PGP to get the inner MIME object, and then it decodes the inner MIME object. Since PGP does not tell the MIME viewer whether the embedded object is MIME or text, the MIME viewer cannot decide the next action for the PGP output. Thus, the MIME viewer must determine the next action before it executes the PGP subprocess.

An open question is here; "Is application/pgp appropriate for conventional PGP/MIME messages which are signed by PGP?". Since it is clear text and MIME view-

```
X-Mew: <1> Good PGP sign "SHIMA Keiichi <keiiti-s@is.aist-nara.ac.jp>" COMPLETE
X-Mew: <2> PGP decrypted.
X-Mew: <2.1> Good PGP sign "SHIMA Keiichi <keiiti-s@is.aist-nara.ac.jp>" COMPLETE
```

Figure 6: PGP/MIME warning

```
ftp://ftp.aist-nara.ac.jp/pub/elisp/Mew/mew-current.tar.gz
```

Figure 7: URL for Mew

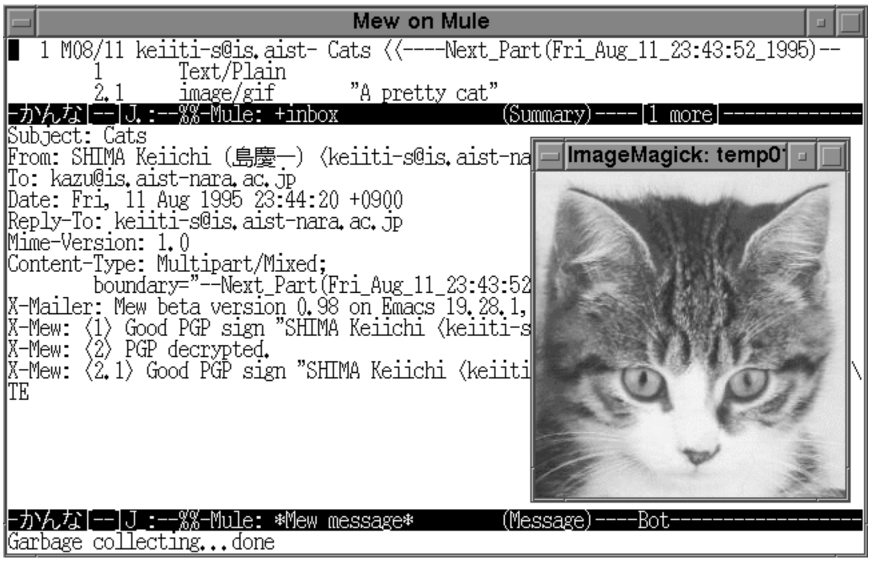

Figure 8: A snapshot of the PGP/MIME viewer

ers treat an object whose subtype of CT: text is unknown as text/plain, "text/pgp" with charset may be proper. We need more experience to fix the spec.

Our PGP/MIME composer achieves our design goals. The composer never defines its composition grammar nor compels users to understand the syntax of PGP/MIME. In fact, Mew users can create any complicated PGP/MIME messages with a simple user manual. The file tree mapping to PGP/MIME syntax is general enough that any other composers on any OS can easily adopt it.

Our PGP/MIME viewer also achieves our design goals. It stores PGP/MIME messages in the format in which they were transported. It caches decoded messages in Emacs buffers so that they can be treated as plain text and be displayed quickly when they are repeatedly read by users. Since the viewer displays the structure of messages, users can enjoy rich operations on any part.

One of the most difficult problems is to forward a PGP/MIME message to a *third* person. Since a PGP/MIME message is encrypted for the receiver, it should be decrypted before forwarding so that the third person can read it. MIME viewers must see if the message includes any encrypted PGP objects then it must decrypt PGP objects and reformat the entire message to obtain a mail-safe form. We believe that a multipart editor can resolve this problem. Since it has not been implemented yet, we do not give further explanation here. Currently, Mew users are compelled to save the decrypted part to a file, then to include the file in a draft buffer.

5 Implementation status and availability

Mew now stably runs on Emacs version 18 and 19, Xemacs, and Mule version 1 and 2. Most of Mew is written by Emacs Lisp, and exceeds 11,000 steps. All features described in this paper have been implemented as well as many other rich functions. We are planning to enhance warning handling during composition, especially for validity of public keys. We are also planning to integrate NetNews to our system. The latest snapshot of Mew is distributed under GNU Public License 2 and is available from the repository showed in Figure 7.

6 Conclusion

This paper described the design and implementation of PGP/MIME. Our PGP/MIME scheme uses the "CT: application/pgp" to enclose a PGP object within MIME and provides the "format" parameter to embed not only localized text but also a MIME object within the PGP object. Those who can decrypt a specific part are not restricted to the receivers of the message.

We implemented a novel PGP/MIME interface, "Mew" on Emacs, which is available as free software. The PGP/MIME viewer of Mew consists of a local form decoder, a syntax analyzer, and a displayer. When a user reads a message, the local form decoder recursively decodes according to CTE:. It also recursively decrypts or verifies PGP objects included in the MIME message, leaving each PGP warning in a corresponding content header. The analyzer analyzes the syntax of the locally formatted message recursively and collects PGP warnings at the same time. Then the displayer simply displays the syntax in an Emacs buffer so that the user can select any part and shows PGP warnings in the mail header. Since decoded messages are cached, the user is never required to enter a passphrase when reading the message again.

Mew provides two methods for composing PGP/MIME. A shortcut method targets localized text that is mostly used in daily life. Users can transform localized text to conventional PGP/MIME with a single key action. The other method is mark based composing, which allows users to create complicated PGP/MIME messages intuitively. Users only need to create a file tree and put marks on each file or directory. Mew converts the directories to multipart and files to singlepart, walking the file tree in postorder. Each part is encoded by PGP according to the marks.

Acknowledgements

The author would like to thank anonymous referees for providing valuable feedback and suggestions. The author is grateful to Atsushi Shionozaki, Darren Stalder, and David Worenklein for reviewing drafts of this paper.

References

[1] D. Crocker, "*Standard for the Format of ARPA Internet Text Message*", RFC822, 1982.

[2] N. Borenstein and N. Freed, "*MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*", RFC1521, 1993.

[3] P. Zimmermann, "*The Official PGP User's Guide*", MIT Press, 1995.

[4] J. Linn, "*Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*", RFC1421, 1993.

[5] S. Crocker, N. Freed, J. Galvin, and S. Murphy, "*MIME Object Security Services*", RFC1848, 1995.

[6] M. Nishikimi, K. Handa, and S. Tomura, "*Mule: MULtilingual Enhancement to GNU Emacs*", Proceedings of INET'93, pp. GAB-1–GAB-9, 1993.