# Mew – 構造を持つメールへのインターフェイス*

山本和彦† 櫻井三子‡ 乃村能成‡‡
† 奈良先端科学技術大学院大学 ‡ 日本電気株式会社 ‡‡ 九州大学

## 概要

文字情報の交換手段として普及したRFC822メールに代わって、本文に構造を持たせることで、音声や画像情報などの複数のメディアを取り込むことができるMIME(多目的メール)がRFC1341で規定されている。しかし、現在ユーザが使用しているMIMEの機能は、漢字などの非アルファベットをヘッダへ挿入することに留まっている。MIMEでは、自分宛の電子メールを関係者に送信(再転送)するための柔軟な形式を定義しているが、頻繁に行われる再転送の多くはMIMEに従っていない。これはMIMEの需要がないのではなく、使いやすいインターフェイスが存在しないことに原因がある。そこで、本稿では電子メールを書く際に広く利用されているエディタEmacs上で実装したMIMEへのインターフェイスMewについて述べる。Mewは、選択的なMIMEの可視器や、ユーザに対してMIMEの文法を隠蔽した構成器によって、使いやすいインターフェイスを提供する。また、PEM(プライバシ強化メール)やPGP(Pretty Good Privacy)等、他の構造化メールとMIMEの統合をMewで実現する方法について議論する。

## 1　はじめに

1982年にRFC(Request For Comment)822[1]で形式が定義されたRFC822メールは、文字情報の交換の手段として世界的に普及した。RFC822メールは、ヘッダと本文を空行で区切るという単純な形式であり、本文には構造を持っていない。

計算機の高速化や大容量化、あるいは、インターネットが英語圏外に普及したことに伴って、文字以外の情報の交換や非アルファベットを積極的にヘッダに挿入する需要が生じてきた。このように電子メールの用途が多目的化するにつれて、さまざまに実装されている電子メールゲートウェイ間で、電子メールを安全に配送できる形式や符号化が必要となった。

そこで、10年を経た1992年にRFC822メールに代わって、RFC1341[2]においてMIME(多目的メール)が規定された。MIMEはRFC822メールの上位互換であると共に、音声、あるいは画像などの文字以外の情報を交換可能である。また、配送の情報が記述されるヘッダに、非アルファベット文字を電子メールゲートウェイが誤動作しない安全な文字列に変換し挿入することができる。

MIMEでは、本文を境界となる文字列でパートという単位に区切り、各パートごとにメディアを取り込むことが可能である。MIMEの構造は階層的に定義されており、複数のパートを持つマルチパートの各パートがシングルパートでもよいし、更にマルチパートを含んでいてもよい。このように本文に構造を持つ電子メールを、本稿では構造化メールと呼ぶ。

現在MIMEの出現から2年が経過したが、ユーザが使用しているMIMEの機能は、漢字などの非アルファベットをヘッダへ挿入することに留まっている。MIMEでは、自分宛の電子メールを関係者に送信(再転送)するための柔軟な形式を定義しているが、頻繁に行われる再転送の

---

* "Mew – An interface to structured mails"

†Kazuhiko YAMAMOTO, Nara Institute of Science and Technology, kazu@is.aist-nara.ac.jp

‡Mine SAKURAI, NEC corporation, m-sakura@ccs.mt.nec.co.jp.

‡‡Yoshinari NOMURA, Kyushu University, nomsun@csce.kyushu-u.ac.jp.

多くはMIMEに従っていない。これは、MIMEの需要がないのではなく、使いやすいインターフェイスが存在しないことに原因がある。

そこで本稿では、使いやすいMIMEインターフェイスについての議論に焦点を置く。まず、既存のMIMEインターフェイスの問題点を明らかにし、MIMEインターフェイスに求められる機能について考察する。そして、これらの要求を満たすMIMEインターフェイスMew(Message interface to Emacs Window)の設計と実装について述べる。このとき、Mewの中心的構成要素であるメールの構造をユーザに見せる可視器と、構造を持つメールを作成させる構成器に分けて議論する。

またMIME以外にも、PEM(プライバシ強化メール)[3]やPGP(Pretty Good Privacy)などの構造化メールが定義され、普及し始めている。これらの構造化メールはそれぞれ独立に発達したために、そのままではこれらを統合することはできない。

現在、PEMやPGPを、構造化メールとして枠組がより広いMIMEに射影するための議論が盛んに行われている。規格が流動的ではあるが、Mewではそれらの規格に基づいてPEMやPGPをMIMEへ統合できることを実証している。本稿ではMIMEの議論だけでなく、構造化メールの統合についてもふれ、構造化メールを一般的かつ柔軟に統合するMewの機能について述べる。

第2節では、既存のMIMEインターフェイスについて考察し問題点を指摘する。そして、MIMEインターフェイスが有すべき機能を第3節で提示する。Mewの可視器は第4節で、Mewの構成器は第5節で説明されている。構造化メールの統合を第6節で議論し、Mewの評価を第7節で与える。最後に、第8節で本稿の結論と今後の課題を示す。

## 2 MIMEインターフェイスの問題点

本節では既存のMIMEに対するインターフェイスを考察し、それらの問題点を指摘する。

### 2.1 前提

現在の代表的なMIMEインターフェイスとして、MH、Metamail、Zmailなどがあるが、すべてコマンドライン上[1]での操作を前提としている[2]。しかしながら、編集や文字入力のためのフロントエンドの理由から、電子メールを読み書きするためにエディタEmacsを用いるユーザが多い。実際、日本語処理を要求される日本人の大半は、Emacs上の代表的な電子メールのインターフェイスであるRmail、mh-e、VMなどを利用している[3]。また、これらのインターフェイスは英語圏でも利用頻度が高い。

これらの理由により、Emacs上での優れたMIMEインターフェイスを提供することが重要であるとの結論に達した。残念ながら、Rmail、mh-e、VMはヘッダの多国語化に対応しているのみであり、本格的なMIMEインターフェイスとはいえない。

### 2.2 可視器の問題点

MIMEの可視化について考察すると、MHに付属するmhnやMetamailでは、MIMEのマルチパートの各パートを上から順に読むことしかできない。Emacsからこれらの MIMEインターフェイスをサブプロセスとして起動した場合は、むろんマルチパートのあるパートを選択的に表示(あるいは印刷、再生など)することはできない。ユーザが本当に望んでいるのは、マルチパートの一部を自由に表示するなどの粒度の細かい操作である。

また、mhnやMetamailは解析したMIME構造を保存しないので、繰り返し同じ電子メールを読むと、その都度構造解析を行う。解析に時間がかかることが分かっているユーザにとっても、同じメールを繰り返し読むときに時間がかかることには、不快感をおぼえるものである。特に解析が必要であることを知らないユーザは、強い不快感を覚えるであろう。

よって、MIMEインターフェイスを実装する際には、各パートを選択的に表示できる可視器を提供し、繰り返し電子メールを読んだ場合に高速に表示することを考慮すべきである。

---

[1] Zmailではコマンドラインの他に、専用のGUIで使用することが可能である。

[2] 著者が知る限り最新である、MHバージョン6.8.3、Metamailバージョン2.7、および、Zmailバージョン3.0を対象とする。

[3] 著者が知る限り最新である、mh-eバージョン4.1、および、VMバージョン5.72を対象とする。

## 2.3 構成器の問題点

第1節で述べたように、MIMEの機能で実際に使われているのは、ヘッダに非アルファベットを使用することのみである。本文の構造化や文字以外の情報の配送などのMIMEの機能はほとんど使われていない。

ユーザは頻繁に電子メールの再転送を行うが、再転送の形式は固定的な境界で電子メールをカプセル化する旧来の書式[4]のままである。MIMEではこの書式を破棄し、境界に任意の文字列を用いることで再帰的な再転送を柔軟に行える形式を規定している。再転送などにMIMEを使用する需要があるにもかかわらず、MIMEが利用されていないのは、使いやすいインターフェイスがないためであると考えられる。

実際に、MIMEのマルチパートを構成する際には、各MIMEインターフェイスに依存した文法を覚えなければならないことが障壁となっている。たとえば、mhnは"#"で始まる複雑な文法を定義しているし、Metamailのmetasendは細かいコマンドラインオプションを覚えなければならない。

ユーザにとって複雑なインターフェイスは、使いにくいばかりでなく間違いを起こしやすい。よって、簡単にMIMEに沿った電子メールを構成できる方法が必要である。多くのユーザがヘッダに非アルファベットを使用しているのは、既存のMIMEインターフェイスが、文字列を自動的に符号化しヘッダに挿入するからである。これは、使いやすいインターフェイスを提供すれば、ユーザが増加することを実証しているといえる。

# 3 インターフェイスに求められる機能

第2節で議論した問題点を踏まえて、本節ではMIMEインターフェイスに求められる機能を示す。

## 3.1 可視器の機能

第2節で述べたように、MIMEインターフェイスには、各パートを選択的に表示、印刷、あるいは、再生するなどの、粒度の細かい操作が必要である。編集や非アルファベットの入力が容易であるEmacsなどのエディタ上のインターフェイスであれば、ユーザにとって利用しやすい。また、同じ電子メールの2回以降の高速表示は、ユーザに不快感を与えないためにも重要である。さらに、さまざまなアプリケーションを取り込むMIMEの潜在的な機能を生かすためには、将来現れるアプリケーションや構造化メールに柔軟に対応できる必要がある。ここで可視器に求められる機能を以下にまとめる。

- 選択性 — 各パートに対する粒度の細かい操作を提供しなければならない。
- 高速性 — 2度目以降の表示は高速でなければならない。
- 柔軟性 — 新しいアプリケーションや構造化メールへ対応できなければならない。

## 3.2 構成器の機能

もし、日々使用する電子メールを書く際に、複雑な操作を要求されれば、ユーザは電子メールを書かなくなるであろう。操作を単純にすることは、ユーザが利用しやすくなるばかりでなく誤操作を冒すことも防止する。よって、MIMEの普及のためには、単純かつ容易なインターフェイスを提供することが重要である。また、今まで会得した知識などを基に直感的に操作を理解できることが必要である。覚えにくい文法を定義すれば、電子メールを書くためにMIMEの文法の他に構成器の文法まで覚えなければならなくなり、直感的というにはほど遠くなる。より優れたインターフェイスに求められることは、ユーザにMIMEの文法を意識させないでMIMEを構成できることである。また、構成できるMIMEの文法に制限があってはならない。たとえば多段のマルチパートやMIMEで取り扱えるさまざまなデータ形式を構成できなくてはならない。そこで構成器に求められる機能として、以下の項目を掲げる。

- 容易性 — 単純な操作でMIMEを構成できなくてはならない。ユーザに複雑な操作を押しつけてはならない。
- 直感性 — 覚えにくい文法を定義してはならない。
- 隠蔽性 — MIMEの文法を理解することをユーザに要求してはならない。
- 汎用性 — 制限無くMIMEが構成できなければならない。

## 4 Mewの可視器

本節では、第3節で提示した要求を満たすために設計したMewの可視器について述べる。まずMIME正規型を定義し、次にMIMEの構造解析について説明する。

### 4.1 MIME正規型

第3節で掲げた高速性を実現するには、(符号化されたデータを)復号し構造解析を行ったMIMEを一時的に保存する必要がある。そこで以下では、保存する際に都合の良い形式について議論する。

ヘッダには電子メールの配送にかかわる情報が保持されているため、電子メールゲートウェイで誤動作の原因となる文字を挿入すべきではない。このため非アルファベットを挿入するには、文献[2]、[5]で定義された符号方式に従って、安全な文字列に変換する。符号化された文字列はそのまま表示しても理解不可能であるから、表示の際には復号する。しかし、電子メールを読むたびに復号を行うのは非効率であるので、一旦復号した電子メールを一時的に保存すべきである。また、Content-Typeの1つの値であるmessage/rfc822は、ヘッダと本文の区別があるテキストを再転送するための型である。このヘッダに符号化された文字列が存在する場合には、同様に復号する必要がある。

ヘッダの議論と同様に、配送を安全にするために符号化されるパートがある。符号方式は、各パートのフィールドContent-Transfer-Encoding:の値に格納される。符号化されたパートは、ヘッダでの理由と同様復号する必要がある。(復号後、このフィールドは無意味となるが削る必要はない。)

たとえば、PEMやPGPなどで電子署名／暗号化を施されたパートは、それぞれのアプリケーションで復号する必要がある。これは、ヘッダでの理由と同様、復号しなければ無意味であるという理由と高速化のためである。復号前のContent-Type:と復号後のパートはデータ形式が違うため、このフィールドを削る必要がある。また、場合によっては、復号後のパートに更にMIME解析を施す必要がある。

これらを考慮して、MIMEを一時的に保存するための形式である**MIME正規型**を定義した。以下にMIME正規型の定義を示す。

**定義 1** *MIME*正規型

- 復号されたヘッダを持つ。また同様に*message/rfc822*のヘッダも復号されている。
- *Content-Transfer-Encoding:*に従って各パートが復号されている。
- アプリケーションで符号化されたデータが復号されており、かつ、対応する*Content-Type:*が削られている。

### 4.2 MIME構造解析

Mewの可視器は、配送後の電子メールを入力しMIME正規型を出力する**MIME復号器**と、MIME正規型に対して構造解析を行う**MIME構造解析器**に分かれる。Mewでは、配送後の電子メールをMIME復号器で処理して得られるMIME正規型を保存する。MewはEmacs上で実装しているので、MIME正規型をEmacsのバッファに保存する。その後、MIME構造解析器はMIME正規型からMIME構造を抽出し、MIME正規型が格納されているバッファに固有な変数として保存する(図1)。

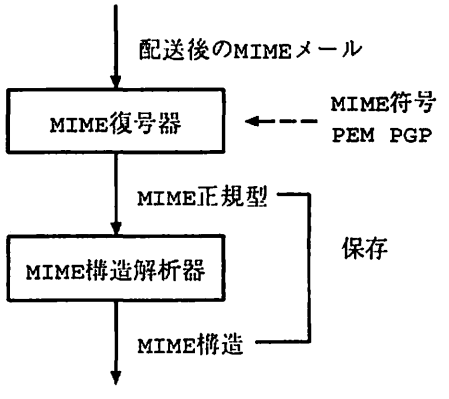


図 1: MIME復号器とMIME構造解析器

Mewにおける**MIME構造**は、あるパートに対する領域とContent-で始まるフィールドの値からなるリスト式で表現される。以下にMewで解析／保存する**MIME構造リスト**の定義を与える。

**定義 2** *MIME*構造リスト

- 各パートはリストで表現される。要素は、領域の開始、終了、*Content-Type:*の値、*Content-Description:*の値、*Content-ID:*の値、*Content-Transfer-Encoding:*の値からなる。ただし、*Content-*で始まるフィールドの値は、主値の他に複数の副値を持つ場合があるので、これらはリストで表現される。
- 第*1*番目の要素がリストの場合、つまり、領域の開始を表す数値でない場合、このパートは全ての要素がパートであるマルチパートを表す。

```
例1
 (910 1020
   ("text/plain" "charset=iso-2022-jp")
   nil nil nil nil)
例2
 (
  (829 1451
    ("text/plain" "charset=iso-2022-jp")
    ("Cover Page") nil nil nil)
  (
   (1663 4853
     ("message/rfc822")
     nil nil nil nil)
  )
 )
```

図 2: MIME構造のリストによる表現

MIMEではContent-で始まるフィールドを予約しており、今後定義2で示した以外にもフィールドが増える可能性がある。しかし、MIME構造リストは要素の数に依存しない構造であるため、容易にフィールド数を追加できる。

図2にMIME構造リストの例を示す。例1では、全体がシングルパートである単純なMIME構造である。領域は910 ~ 1020[4]であり、Content-Type:の値としてtext/plain、副値としてcharset=iso-2022-jpが格納されている。他のフィールドは存在しないので、nilとなっている。例2は、マルチパートの例である。第1番目のパートはContent-Type:がtext/plainであり、Content-Description:としてCover Pageという値がある。また、第2番目のパートは、Content-Type:としてmessage/rfc822を持つ1つのシングルパートからなるマルチパートである[5]。

このようにMewでは、MIME正規型とMIME構造リストを保存するため、2回目以降に電子メールを読む場合に、復号の手間を省略しヘッダや各パートを高速に表示することが可能である。また、MIME復号器とMIME構造解析器を分離したことによる柔軟性については第6節で述べる。

## 4.3 MIME復号器と構造解析器

MIMEのパートは階層的に構成することができる。たとえば、あるパートがマルチパートを含み、また、そのマルチパートのいくつかのパートがマルチパートであってもよい。よって、MIME復号器は再帰的にMIME正規型を作る必要があり、また、MIME構造解析器は再帰的にMIME構造を抽出しなければならない。Mewにおいて両者の実装は本質的に同一であるので、以下MIME構造解析器の動作についてのみ説明を行う。

Mewではある1つのパートを解析する関数$S$とマルチパートを解析する関数$M$を実装している。MIME正規型は、それ全体で単一のパートであるので、まず$S$で評価される。$S$では、Content-Typeを調べマルチパートである場合は$M$を呼ぶ。シングルパートの場合は、各フィールドの値をリストにして、そのパートのMIME構造リストとして返す。$M$では、マルチパートに含まれるそれぞれのパートに対して、$S$を呼び出す。このように$S$と$M$は互いに再帰的に呼び合い、MIME構造を抽出する。

MIME構造解析器の解析時間は、ほぼMIME正規型の長さに比例する。MIME復号器の処理時間は、復号した後のパートをさらに復号する可能性があるため、MIMEの構成に依存する。

図3では、第1番目のパートがtext/plain、第2番目のパートがaudio/basicとimage/gifのマルチパートからなるマルチパートの解析を示している。

[4] バイト数と考えてよい。

[5] 括弧、つまり、パートだけに注目すると((()(())))となっていることが分かる。

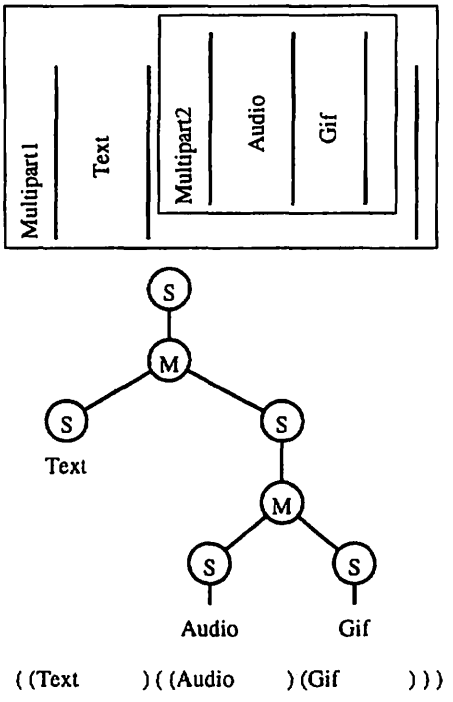


図 3: 関数 $S$ と $M$ による再帰的な解析

### 4.4 可視化

Mew ではフォルダ[6]に格納されている電子メールを、図 4で示すように一覧表示する。ここで、(図中では * で表されている) カーソルの下にある電子メールを「現在の電子メール」と呼ぶこととする。

ユーザが表示コマンドを入力した場合、Mew は現在の電子メールをバッファ内に読み込み、MIME 復号器と MIME 構造解析器にかける。もし、電子メールがシングルパートの場合は、即座に Content-Type: の値に対応した表示を行う。たとえば、text/plain は他の Emacs ウインドウに表示し、application/postscript は ghostview などのコマンドを起動する。

もし、現在の電子メールがマルチパートの場合は、図 5で示すように、一旦 MIME 構造を可視化する。このため、ユーザはカーソルを動かし表示のコマンドを入力することで、各パートを選択的に表示することが可能である。

Mew では、Content-Type: の値に応じて呼び出すコマンドや Lisp 関数を、ユーザが自由に設定できる。

[6]Mew では電子メールを格納する形式として、MH のフォルダを利用している。電子メールは数字のファイル名で保存される。

## 5 Mew の構成器

本節では、第 3節で提示した要求を満たすために設計した Mew の構成器について述べる。

### 5.1 Content-Type: の取り扱い

ユーザが実際に利用する MIME は、シングルパートか、1 段あるいは 2 段のマルチパートのように単純な構造である場合が多いことが分かっている。しかしながら、構成器はユーザの利用方法を限定すべきではないので、多段のマルチパートを構成できるように設計することが望ましい。

マルチパートの枠を用意してから、内部の各パートを編集するトップダウン的な構成を行いたいと思うユーザもいるし、シングルパートを編集してからマルチパートの 1 つのパートに挿入するボトムアップ的な構成方法を要求するユーザも考えられる。このため、Mew ではトップダウン的にも、ボトムアップ的にも MIME を構成できるようにすることを目指した。

前述のように、各パートにはデータ形式を表す Content-Type: が付加される。トップダウン的にもボトムアップ的にも MIME を構成できる構成器において取り扱いが難しいのは、ヘッダに挿入する Content-Type: を決定することである。つまり、構成方法に制限がないため、一番外側の Content-Type: は電子メールの送信直前まで決定できない。

そこで、Mew では MIME を構成する方法として、「電子メールを送る直前に本文の一番外側の Content-Type: ヘッダに挿入する」という方法をとっている。Mew では、各パートの Content-Type: が自動的に付加されるため、ユーザは入力する必要がない。これについては、次小節で詳しく述べる。

電子メールを書く際に、Mew ではヘッダと本文と文字列 "----" で区切ったテンプレートが自動的に用意される。電子メールを作成している例を図 6に示す。ここで、この平文をユーザが送信

```
 547 M07/18 Yoshinari NOMURA  Re: summary only mode (( のむらです。 ) いまからの
*548 M07/18 Mine Sakurai (m-s mew-url 0.09 ((---- Outer Boundary (Mon Jul 18 20
 549 M07/19 Kazumasa Utashiro Re: 0.54 ((Apparently-To: のアドレスが mew-summar
```

図 4: 電子メールの一覧

```
*548 M07/18 Mine Sakurai (m-s mew-url 0.09 ((---- Outer Boundary (Mon Jul 18 20
        1       text/plain
        2.1     text/plain
 549 M07/19 Kazumasa Utashiro Re: 0.54 ((Apparently-To: のアドレスが mew-summar
```

図 5: MIME 構造の表示

```
To: mine
Subject: PEM を使おうよ。
Mime-Version: 1.0
----
Mew が PEM をサポートしました。
テストしてみてね。

-- かず
```

図 6: 平文

```
To: mine
Subject: PEM を使おうよ。
Mime-Version: 1.0
----
Content-Type: application/pem-1421

-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type:4,ENCRIPTED
                          以下省略
```

図 7: 暗号化した本文

しようとすると、Mew はデフォルトの Content-Type: として text/plain をヘッダに挿入する。(つまり、ヘッダの直後に Content-Type: がない場合は、text/plain が省略されているように扱う。) ユーザが平文を暗号化するために PEM の処理を施すと、Mew は本文の一番外側に PEM を表す Content-Type: を挿入する (図 7)。そして、電子メールの送信直前にヘッダの直後にある Content-Type: をヘッダに挿入する。

このように Mew では、電子メールの送信の直前にヘッダの Content-Type: を決定することで、ユーザに自由な構成方法を提供している。

## 5.2 マルチパートの構成方法

Mew では直感的にマルチパートを構成できるように、ディレクトリ構造を MIME 構造へ射影する機能を持っている。つまり、ディレクトリはマルチパートに、ファイルはシングルパートに変換される。また、各ファイルの Content-Type: は、ファイル名から推測できる[7]。たとえば、拡張子 ".ps" を持つファイルは application/postscript であるし、"11" のようにファイル名が数字であるファイルは message/rfc822 であると推測できる。

UNIX をある程度使った経験があるユーザであれば、ファイルの複製、移動、リンク、および、削除や、拡張子などのファイル名の慣習は理解できている。また、ディレクトリは階層化することが容易である。

Mew でマルパートを作成している例を、図 8に示す。ここで、マルチパートを構成するためのコマンドを入力すると、本文の下に、文字列

```
---- multipart --
```

と

```
---- multipart ----
```

で囲まれた、マルチパート部が追加される。マルチパート部では、ファイルの複製、移動、リンク、削除や、ディレクトリの生成、消去などのコマンドが各キーに割り当てられている。図 8は、MIME を構成するためのディレクトリに、ファイル 1、cat.gif、mew.patch をコピーした例で

[7] 著者の知る限り、mime.el がファイル名から Content-Type: を推測する機能を最初に提供した。

ある。ここで、ディレクトリ構造をマルチパートへ射影するコマンドを入力すれば、MIMEができあがる。

```
To: mew-dist
Subject: ここがヘッダ
Mime-Version: 1.0
----

* ここは本文です。

---- multipart --
drafts/mime/1/
    1            message/rfc822
    cat.gi       image/gif
    mew.patch    text/plain
---- multipart ----
```

図 8: マルチパートの構成

このようにMewでは、MIMEの文法を見せずに、ユーザに分かりやすいディレクトリやファイルを用いることで、非常に使いやすいインターフェイスを与えている。

## 6 MIME以外の構造化メールの統合

本節では、構造化メールであるPEMやPGPをMIMEへ統合する方法について議論する。

### 6.1 PEM

RFC1421[3]で規定されているPEM(RFC1421 PEM)では、本文に電子署名を施した場合、さらに暗号化まで行った場合、電子メールの本文において、結果を

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
```

と

```
-----END PRIVACY-ENHANCED MESSAGE-----
```

で囲む。このように、PEMは構造を持った電子メールであるにもかかわらず、新しくフィールドを定義していない。そのため、電子メールがPEMであるかを検査するには、本文に対して文字列検索を行わなければならない。

そのため、長い電子メールでは非効率的であり、また、同じ文字列を含む平文と明確に区別できない。そこで、本文がPEMであることを示す新しいフィールドを定義する必要がある。つまり、MIMEのContent-Type:フィールドに、PEMであることを示す値を定義すればよい。

1993年10月時点では、PEMとMIMEを統合する方式として、文献[6]で定められているマルチパートを使う方法と、文献[7]で規定されているapplication/pem-1421を使う方法があった。現在では、マルチパートを使う方法へ一本化されている。

Content-Type:の値として、application/pem-1421を使う方法では、RFC1421 PEMをそのまま利用することができる。マルチパートを使う方法では、RFC1421 PEMよりも機能は上がっているが互換性はない。

我々は最終的にはマルチパートを使う方式をサポートする予定である。しかし、既存のRFC1421 PEMの需要があるため、一時的な解としてapplication/pem-1421をContent-Type:の値として利用している。これにより、ヘッダを検索するだけで本文がPEMであるか否かを判断できる。

Mewの実装を通じて分かったことは、PEMで復号したパートの解釈の仕方が、復号前に分かっていなければならないことである。つまり、PEM用のContent-Type:には、PEMで電子署名／暗号化する前の本文が、RFC822メールであるのかMIMEであるのかを示す情報がなければならない。なぜなら、たとえばEmacsからPEMのコマンドを呼び出し、復号された結果を受け取った際に、Emacs側で次に行う動作が決定できなければならないからである。

文献[7]では、復号後の解釈が与えられていない。そこで、現在Mewでは、Content-Type:としてapplication/pem-1421を値とする電子メールは、復号後にRFC822メールとして解釈するように実装している。

### 6.2 PGP

PEMとまったく同様に、PGPでも、本文がPGPで処理されていることを示すフィールドがヘッダに定義されていない。文献[8]では、Content-Type:の値としてapplication/pgpを定義し、さらに、復号後の処理を副値で与えている。

これは文献[7]と本質的に同じ方法であるため、PEMでこの提案が破棄された現在、良い規格とは言いがたい。また、文献[6]の方法は、

PGP を統合する枠組を持っているので、文献 [6] の方法に従って PGP と PEM が協調的に仕様を決定していくのがよいであろう。

このような理由から、現在 Mew では、RFC822 メールに対してのみ PGP の処理ができるように実装を行っている。application/pgp を Content-Type: として利用しているが、復号後の処理を表す副値は利用していない。

### 6.3 MIME 正規型による柔軟性

これまで考察してきたように、MIME と他の構造化メールを完全に統合するには、適切な Content-Type: を定め、Content-Type: に対応するアプリケーションで処理した結果を解釈するための情報が、その Content-Type: の値内に与えられていればよい。

前述のように、Mew では MIME 正規型の定義により、到着後の電子メールを MIME 復号器と MIME 構造解析器によって処理する。新たな構造化メールが出現した場合は、MIME 復号器を修正することによって対応することができる。つまり、MIME 構造解析器以降の処理には一切変更を必要としない。このように、Mew の可視器は新しい構造化メールに対して柔軟に対応できるように設計されている。

## 7 比較、評価

本節では Mew と他の MIME インターフェイスである Zmail、Metamail、mhn、および、mime.el とを比較する。それぞれ、可視器と構成器について評価するが、MIME を構成するための mime.el は、可視器の機能はないので構成器についてのみ考察する。

### 7.1 可視器の比較

可視器において選択性を有している MIME インターフェイスは、Mew と Zmail である。Mew はカーソルを移動させて表示コマンドを入力することで、Zmail はアイコンをクリックすることで各パートを選択できる。Metamail や mhn では順番通りにしか読み出せない。

高速性や柔軟性を実現している MIME インターフェイスは、MIME 正規型を使用している Mew のみである。

表 1: 可視器の評価

| | 選択性 | 高速性 | 柔軟性 |
|---|---|---|---|
| Mew | ○ | ○ | ○ |
| Zmail | ○ | × | × |
| Metamail | × | × | × |
| mhn | × | × | × |

### 7.2 構成器の比較

構成器において、容易性を実現できているインターフェイスは Mew と Zmail である。Mew では、ファイルやディレクトリを扱う各コマンドを一文字のキーに割り当てており、また、誤動作に対する訂正が簡単である。Zmail でも、ファイルを取り扱うだけで MIME が構成できる。ただし、Zmail での構成器はファイルのブラウザと協調的に動作しないので、ユーザはファイルへのパスを入力しなければならない。可視器がアイコンを利用しているだけに、構成器においてアイコンをドラッグ & ドロップするだけで MIME が構成できないのは統一感に欠ける。Metamail は複雑な引数を指定しなければならず、また、mhn は複雑な文法を入力する必要があるので容易性に欠ける。mime.el では、誤動作を起こしにくいように、コマンドがキーに割り当てられておらず、また、誤動作の訂正が行いにくい。

全ての MIME インターフェイスがファイルをシングルパートに対応させている。ディレクトリをマルチパートへ対応付けているインターフェイスは Mew だけである。よって、Mew のみが直感性を有しているといえる。

Mew、Zmail、および、mime.el は、独自の文法を定義しておらず、また、MIME の文法さえ隠蔽することに成功している。Metamail では独自の複雑なコマンドラインオプションを理解する必要があるし、mhn では固有の複雑な文法を覚えなければならない。

MIME を制限なく構成でき、さまざまな構造化メールを取り扱えるのは Mew のみである。他の構成器は、さまざまな構造化メールに対応していない。また、Zmail と mime.el は多段のマルチパートを構成できない。

表 2: 構成器の評価

| | 容易性 | 直感性 | 隠蔽性 | 汎用性 |
|---|---|---|---|---|
| Mew | ○ | ○ | ○ | ○ |
| Zmail | ○ | × | ○ | × |
| Metamail | × | × | × | × |
| mhn | × | × | × | × |
| mime.el | × | × | ○ | × |

### 7.3 Mew の評価

第3節で掲げた可視器への要求である選択性、高速性、および、柔軟性、また、構成器への要求である容易性、直感性、隠蔽性、および、汎用性を Mew は完全に実現している。また、現在 Mew は、複数の構造化メールを取り扱うことのできる唯一のインターフェイスである。

Mew の配布前までは、筆者はマルチパートで構成される MIME をほとんど受け取ることがなかった。しかし、Mew の配布後は比較にならないほどマルチパートの MIME を受け取るようになったことを、統計的な評価ではないが書き添えておく。

## 8 おわりに

本稿では、現在の MIME インターフェイスを考察し、可視器には選択性と高速性、柔軟性が、構成器には容易性、直感性、隠蔽性、および、汎用性が必要であると述べた。我々が実装を行っている Mew の可視器は、MIME 復号器と MIME 構造解析器により高速かつ柔軟である。また、MIME の各パートに対して選択的な操作を行える。Mew の構成器は、ディレクトリ構造を MIME 構造へ射影する機能を持っている。そのため、ユーザはファイル名の慣習やディレクトリ構造から MIME 構造を直感的に理解でき、また、ファイルやディレクトリに対する単純な操作を行うことで、MIME を制限無く構成できる。このように MIME の文法をユーザから完全に隠蔽していることは特筆すべきことである。

仕様が流動的ではあるが、PEM や PGP のように本文に構造を持つ電子メールを、構造化メールとして、より一般的な枠組である MIME に統合できることを Mew は実証している。仕様が確定次第、構造化メールの統合を本格的に実装する予定である。また、今後 Mew でネットニュースへのインターフェイスを提供することを計画している。


## 謝辞

WIDE セキュリティ分科会とマルチメディア分科会のメンバーには、MIME と PEM の統合に関し議論して頂いた。また、機能が不足していた初期の Mew から根気強くテストを行って下さったのは、Mew メーリングリストのメンバーである。特に歌代和正氏は可視器を単純にすることを示唆して下さり、門林雄基氏にはディレクトリ構造をマルチパートへ射影する構成器のアイディアを提供して頂いた。ここに記して心から感謝する。


## 参考文献


[1] D. Crocker, "Standard for the format of ARPA Internet text messages", RFC822, August 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", RFC 1521, September 1993

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

[4] Marshall T. Rose and Einar A. Stefferud, "Proposed Standard for Message Encapsulation", RFC934, January 1985

[5] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text", RFC 1522, September 1993.

[6] Steve Crocker, Ned Freed, Jim Galvin, and Sandy Murphy, "PEM Security Services and MIME", working draft, July 1994.

[7] J. I. Schiller, "An Alternative PEM MIME Integration", working draft(obsoleted), October 1993.

[8] N. Borenstein, C. Plumb, and P. Zimmermann, "The application/pgp MIME Content-type", working draft, May 1994.