百万人のMutt
～Mutt活用講座～

滝澤隆史
taki@cyber.email.ne.jp

第4回　PGP

今回はPGPについて説明を行います。読者の中には「PGPって何ですか？」という人もいると思うので、前半ではPGPについて簡単に紹介し、PGPのフリーな実装であるGnuPGのインストール及び設定に関して説明を行います。後半ではMuttでPGPを使う方法を紹介します。

本題に入る前に、Muttの最新状況についてお知らせしておきましょう。執筆時点での最新バージョンは1.3.20です。1.3.19からの大きな変更点は、設定コマンド「account-hook」が追加されたことです。このコマンドは、複数のPOP/IMAPのアカウントを使う場合に役に立ちます。なおこの1.3.20では、./configureのオプション「--with-iconv=DIR」が「--with-libiconv-prefix=DIR」に変わっています。インストールを行う際には注意するようにしてください。

■用語解説

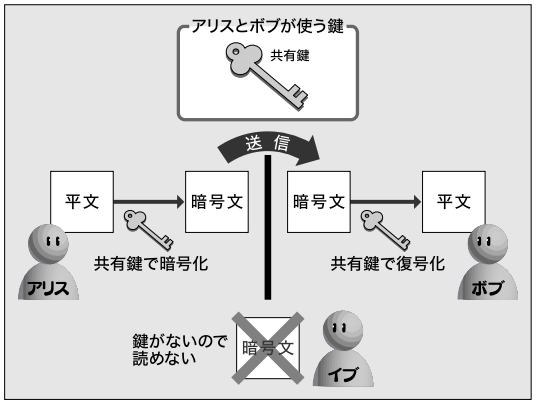
まず、この記事を読むに当たって最低限必要だと思われる用語を簡単に説明します。さらに詳しく知りたい方は、参考書籍とサイトをまとめましたので、記事末のResourcs【1】～【4】をご覧ください。

●共有鍵暗号化法

「共有鍵暗号化法」とは、暗号化と復号化をできる同一の鍵を当事者間で共有する方法です。

例を示すと、アリスが鍵を使って暗号化した文書をボブに送ったときに、ボブは同じ鍵で復元してその文書を読むことができます。途中でイブが暗号文を手に入れても鍵を持っていないので読むことはできません。ただし、アリスとボブは何らかの安全な方法であらかじめ鍵の受け渡しを行う必要があります（図1）。

【図1】共有鍵暗号化法



共有鍵暗号化法は大量のデータ送信などに使用されます。

●公開鍵暗号化法

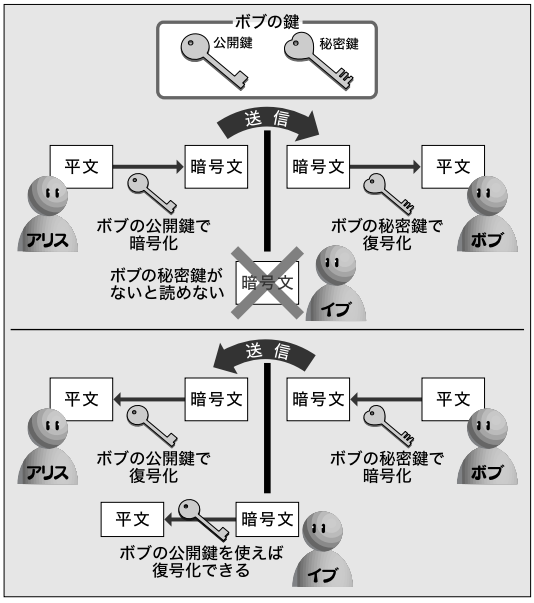
「公開鍵暗号化法」とは、「公開鍵」と「秘密鍵」という2つの鍵の組み合わせで暗号化と復号化を行う方法です。公開鍵で暗号化した文書は秘密鍵で復号化でき、秘密鍵で暗号化したものは公開鍵で復号化できます。公開鍵で暗号化した文書は公開鍵では復号化できないため、暗号通信を行うためにはお互いに公開鍵を公開しさえすればよいことになります。

例を示すと、アリスがボブの公開鍵で暗号化した文書をボブに送ると、ボブは自分の秘密鍵でその文書を復号化して読むことができます。途中でイブが暗号文を入手しても、ボブの秘密鍵を持っていなければ読むことはできません。このように、ボブに暗号文を送りたい人は、公開されているボブの公開鍵を入手して使用します。

逆に、ボブの秘密鍵で暗号化した文書は、ボブの公開鍵でしか復号化できません。これはその文書をボブが暗号化したことが証明されていることになります（図2）。

なお、「公開鍵暗号化法」は「共有鍵暗号化法」に比べて暗号化と復号化に計算時間がかかります。そのため実際には「共有鍵暗号化法」と組み合わせて用いられています。

【図2】公開鍵暗号化法



公開鍵暗号化法は、共有鍵の送信や電子署名などに使われます。

●電子（デジタル）署名

「電子（デジタル）署名」とは、メッセージを送ってきた人が本人であることを証明するものです。これにはメッセージのハッシュ値[*1]を秘密鍵で暗号化したものが使われています。

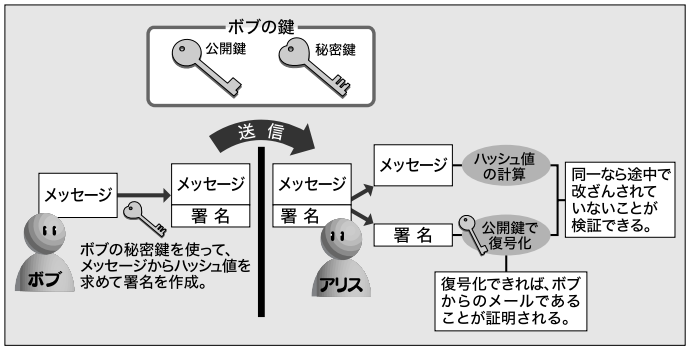
電子署名をメッセージに付けて送ると、受け取られたメッセージの署名は、その署名が送信者の公開鍵で復号化できることで、本人から送られたことが証明されます。また、メッセージのハッシュ値と、署名を復号化したハッシュ値が同一なら、メッセージが途中で改ざんされていないことが検証できます（図3）。

■PGPの概要

PGP（Pretty Good Privacy）はPhilip R. Zimmermann氏によって開発され、1991年に

＊1　電子（デジタル）署名を作成するために、メッセージダイジェスト関数を使用して導き出される値のこと。メッセージごとに全く異なる値が導き出されます。メッセージダイジェスト関数は逆変換が行なえないことから「一方向ハッシュ関数」とも呼ばれます。

【図3】電子（デジタル）署名



【実行例2】公開鍵サーバへの登録

```
$ gpg --send-key
gpg: success sending to 'pgp.nic.ad.jp' (status=200)
```

公開された暗号化／電子署名を行うソフトウェアです。公開後にさまざまな人の手が加わり、また、使用しているアルゴリズムのライセンスの関係による紆余曲折を経て、ライセンス面でクリアになったPGP 2.6*2が出てから、よく使われるようになりました*3。このバージョンの形式に関してはRFC 1991「PGP Message Exchange Formats」として公開されています。

PGP 2.6は、RSAやIDEAなど特定企業の特許の制限を受けるアルゴリズムを用いていましたが、PGP 5.0では特定企業の特許に影響を受けない公開鍵暗号として署名用に「DSS(DSA)」を、暗号用に「Diffie-Hellman（実体はElGamal）」を採用し、共有鍵暗号として「Triple-DESとCAST5」を採用し、ハッシュアルゴリズムとして「SHA-1」を採用しました。IETF*4は、Network Associates Inc.（旧PGP Inc.［5］）の協力のもと、このPGP 5.0を参考にして新しいPGPとして「OpenPGP」を定め、RFC2440「OpenPGP Message Format」として公開しました。

現在、PGP 5.0の後継としてPGP 6.5.8およびPGP7.0.3（UNIX版はない）がNAIから商用のソフトウェアとして販売されていて、非商用の個人的な使用に限りフリーソフトウェア版が使用できます。ソースコードも公開されています。また、完全にフリーな実装として「GnuPG（GNU Privacy Guard）」（［6］）がWerner Koch氏を中心として開発されていて、1.0.6が公開されています。こちらは商用、非商用にかかわらず自由に使用できます。

■PGP/MIME

MIMEに統合する前のPGPの欠点として、文字符号化方式の指定ができない、データの符号化を指定できない、「-」で始まる行の扱い、などいくつかの問題を抱えていました。そのため、PGPをMIMEの枠組みの中で扱えるようにしたのが「PGP/MIME」です。RFC2015「MIME Security with Pretty Good Privacy（PGP）」として公開されています。これにより、メッセージパートと署名パートとが明確に分かれ、先に挙げた問題は解決し、また、暗号化と署名との分離なども行われるようになりました。

ちなみに、このRFCの著者Michael Elkins氏は我らがMuttの作者です。RFC2015はPGP 2.6を前提として書かれているので、OpenPGPをベースにした改定案「MIME Security with OpenPGP（OpenPGP/MIME）」が現在議論されています。この改定案の著者グループには先のElkins氏に加え、現在のMuttのコミッターであるThomas Roessler氏も入っています。

■GnuPG

ここではGnuPGのインストールと設定について説明しましょう。

●インストール

配布元のFTPサイト（［7］）*5から直接入手してもよいですが、Ring Server（［8］）にミラーされているのでそちらから入手してください。最新バージョンは1.0.6です。過去のバージョンはバグやセキュリティホールの問題があるので、最新バージョンを使うようにしてください。

ダウンロードが完了したら、GnuPGのサイト（［6］）の「Download」ページに示されているMD5チェックサムの値と、次のコマンドの結果と比較し、一致しているか確認します。

```
$ md5sum gnupg-1.0.6.tar.gz
```

確認できたら、展開してコンパイルします。

```
$ gzip -dc gnupg-1.0.6.tar.gz |
tar xvf -
$ cd gnupg-1.0.6
$ ./configure
$ make
```

rootで「make install」を実行してインストールを行い、最後にメモリページをロックするためにgpgをrootのSUIDにします。

```
# chmod 4755 /usr/local/bin/gpg
```

これでGnuPGのインストールは完了です。

●鍵の生成

実行例1に鍵の生成過程を示します。

●~/.gnupg/optionsの設定

gpgの各種設定は~/.gnupg/optionsで行います。

まず、起動するたびに表示されるコピーライトが表示されないようにします。次の1行を~/.gnupg/optionsに追加してください。

```
no-greeting
```

NAIのPGP 6.5以前では、OpenPGPで定義されているV4形式の署名を扱えません。そのため、他のPGPの実装との相互運用性のために、V3形式の署名を強制的に作成するオプションを追加します。このオプションは標準で記述されているはずですから、確認しておいてください。

```
force-v3-sigs
```

続いて「公開鍵サーバ」を指定します。これを指定することにより、必要なとき自動的に公開鍵を取得してくれるようになります。公開鍵サーバは「host -l pgp.net | grep www」の出力結果から、ネットワーク的に近いと思われるところを選んでください。このリストには入っていませんが、日本国内の場合は「pgp.nic.ad.jp」を指定するといいでしょう。

```
keyserver pgp.nic.ad.jp
```

以上の設定が完了したら、先ほど生成した公開鍵を次のコマンドを実行して、公開鍵サーバに登録します（実行例2）。

●PGP 2.6との互換性

GnuPGは、特許で制限のあるアルゴリズムは一切使っていませんし、また、使わない方針でもあります。そのため、PGP 2.6で用いられているRSAとIDEAは特許に抵触するため、GnuPGでは公式にはサポートしていませんでした。しかし、RSAに関しては2000年に特許が切れたので、1.0.3から配布パッケージに含まれています。

一方IDEAに関しては、プラグインとして組

＊2　RSA Data Security Inc.のRSAREF(TM)ツールキットを組み込むことによって、RSAREF(TM)と同じライセンスが適応されるようになりました。
＊3　書籍「PGP 暗号メールと電子署名」（記事末のResource［1］を参照）に詳しい経緯が書いてあります。
＊4　The Internet Engineering Task Force。インターネットのさまざまな決まり事を議論して決める団体。
＊5　ちなみにMuttの配布も同じFTPサイトで行っています。

【実行例1】鍵の生成過程

```
$ gpg
gpg: /home/taki/.gnupg: directory created
gpg: /home/taki/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
$ gpg --gen-key
gpg (GnuPG) 1.0.6; Copyright (C) 2001 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: /home/taki/.gnupg/secring.gpg: keyring created
gpg: /home/taki/.gnupg/pubring.gpg: keyring created
Please select what kind of key you want:
      (1) DSA and ElGamal (default)
      (2) DSA (sign only)
      (4) ElGamal (sign and encrypt)
   Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
            minimum keysize is  768 bits
            default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024) 2048
Do you really need such a large keysize? y
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
   Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs the user id
from Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Foo Bar
Email address: foo@example.org
Comment:
You selected this USER-ID:
    "Foo Bar <foo@example.org>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

Enter passphrase:
Repeat passphrase:
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
.++++++++++++++++++++++++.+++++++++++++++.+++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++.+++++++++++++++.++++++++++++++..+++++>+++++.....
..............++.++
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++..+++++.+++++++++++++++..+++++..+++++..++++++++++.+++++++++++++++.+++++
.+++++.++++.++++++++++.++++++++++.++++++++++.+++++..+++++.+++++..+++++++++++
++++.+++++.+++++>+++++..+++++.>+++++....+++++^^^^^^^^^
public and secret key created and signed.
```

- $ gpg：初めてgpgを実行するとディレクトリ~/.gnupgと設定ファイル~/.gnupg/optionsが生成されます。
- $ gpg --gen-key：--gen-keyオプションを付けてgpgを実行すると、「鍵リング」が作成され、いくつかの質問が行われます。
- Your selection? 1：生成する鍵の種類を選びます。ここではデフォルトの(1)を選択します。
- 「DSA鍵ペア」は1024ビット固定です。「ELG-E鍵ペア」の場合には、鍵の強度を高めるために2048を入力します。そのかわり計算に時間がかかるようになりますが、最近のマシンでは問題ないでしょう。
- Do you really need such a large keysize? y：確認メッセージが表示されたら「y」を入力します。
- Key is valid for? (0) 0：鍵の有効期限は無期限として「0」を入力します。
- Is this correct (y/n)? y：確認メッセージが表示されたら「y」を入力します。
- 「実名」と「コメント」と「メールアドレス」を入力しユーザIDを作成します。ここで漢字は入力しないでください。
- "Foo Bar <foo@example.org>"：ユーザーIDが作成できました。
- Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o：表示されたユーザーIDでよければ「o」を入力します。
- Enter passphrase: / Repeat passphrase:：秘密鍵のパスフレーズを入力します。
- 乱数を生成するためにキーボードを押したりマウスを動かします。
- public and secret key created and signed.：このメッセージが表示されたら鍵の生成は完了です。

＊6　このディレクトリのパスはご使用の環境によって異なる場合があります。

み込める非公式モジュール「idea.c」が公開されていましたが、現在は公開を取りやめています。ただし、idea.cを配布しているサイト（[9] など）はまだあるので、そちらから入手することができます。なお、使用に当たってはライセンスに注意してください。非商用利用で個人間のデータ転送に使う場合に限って特許権使用料はかからないことになっています。

コンパイルおよびインストール方法は次の通りです。

```
$ gcc -Wall -O2 -shared -fPIC -o
idea idea.c
# cp idea /usr/local/lib/gnupg/
```

次に~/.gnupg/optionsに次の1行を追加してください。

```
load-extension idea
```

■ Mutt と PGP

● PGP への対応

現在UNIX系OSで使われているPGPの実装には、PGP 2.6、PGP 5.0、PGP 6.5、GnuPGがあり、それぞれ使い方や動作が異なります。そのため、メーラがPGPに対応しようとしたら、それぞれ個別の実装ごとに対応しなければなりません。Muttではその実装の違いを吸収するために、署名の検証、暗号化などの動作に応じたPGPのコマンドを、設定変数で指定できるようにしています。

具体的な例を示すと、GnuPGで署名の検証をする場合は、設定変数$pgp_verify_commandに署名の検証を行うコマンド「gpg --no-verbose --batch --output - --verify %s %f」を指定します（リスト1）。このようにすれば、署名の検証を行うときにPGPのバージョンや実装の違いを気にせずに$pgp_verify_commandに指定したコマンドを単に実行するだけで済みます。

Muttにはこのようなラッパー変数がPGPの機能ごとにありますが、ユーザーが個別に設定するのは面倒です。そこで、バージョンや実装ごとに設定変数を記述した設定ファイルが/usr/local/doc/mutt/samples/*6に用意されています。PGPの実装に応じた設定ファイル名（例えばGnuPGならgpg.rc）のファイルを読み込むようにします。読み込み方法については次ページの「PGPを使うための準備」を参照してください。

● PGP/MIME と旧形式の PGP

Muttは、「PGP/MIME」とMIMEでない「旧形式のPGP」の両方に対応しています。しかし、旧形式のPGPに関してはその使用が推奨され

ていないため、いくつかの制限があります。

まず、旧形式のPGPメッセージの作成に関してですが、デフォルトでは無効になっています。旧形式のPGPメッセージを作成するには、環境変数「$pgp_create_traditional」を次のように設定します。

```
set pgp_create_traditional=yes
```

ところがこの変数を設定しても、「US-ASCII」以外の文字符号化方式の場合は、自動的にPGP/MIMEになります。そのため日本語版では、「ISO-2022-JP」の場合には自動的にPGP/MIMEにならないようにしています。

旧形式のPGPとはいっても、Muttの場合は純粋な旧形式ではなく、中途半端なMIME化が行われています。そのため、本文は旧形式になるものの、ヘッダにはリスト2に示すような一文が追加されます。Content-Typeが「text/plain」でないため、相手によってはうまく認識できない場合があるかもしれません。

旧形式のPGPメッセージの表示に関しては、文字符号化方式の変換が行われないため、変換が必要となる日本語のメッセージは表示できません。これでは困るので、日本語版では変換を行うようにして表示させています。

結論としては、「旧形式のPGPメッセージを扱うことはできるが、送るのはやめた方がよい」ということになります。要は「$pgp_create_traditional」を設定しなければよいだけです。

■PGPを使うための準備

まず、PGPに対応した設定ファイルをMuttの設定ファイルに読み込みます。GnuPGを使う場合は、リスト3に示す記述を加えます（このディレクトリのパスはご使用の環境に合わせて変更してください）。その後MuttでPGPが使えるかどうかを次のコマンドで確認します。

```
$ mutt -v | grep PGP
```

「+HAVE_PGP」という文字列が表示されればPGPが使えます。「-HAVE_PGP」と表示されたら、そのMuttではPGPを使うことはできません。後者の場合、Muttをインストールするときに、「--enable-pgp」オプションを付けずに./configureを実行した可能性があります。この場合には、実行例3のようにして再インストールを行ってください。

なお、Muttをインストールするより前にPGPがインストールされている場合には、PGPの使用が自動認識されるため、--enable-pgpオプションを明示してインストールする必要はありません。

【リスト1】環境変数$pgp_verify_commandの指定例

```
set pgp_verify_command="gpg --no-verbose --batch --output - --verify %s %f"
```

【リスト2】Muttで旧形式PGPを使用したときに付けられるヘッダ

```
Content-Type: application/pgp; x-action=sign; format=text
```

【リスト3】Muttの設定ファイルに追加する内容（GnuPGの場合）

```
source /usr/local/doc/mutt/samples/gpg.rc
```

【表1】PGP関連の設定変数

環境変数	処理	デフォルトの値
pgp_good_sig	署名の検証時の出力結果に$pgp_good_sigを含んでいたら成功と見なす	""
pgp_retainable_sigs	署名してから暗号化する場合に、マルチパート構造の署名メッセージを入れ子にして暗号化する	no
pgp_strict_enc	署名するメッセージを自動的にquoted-printableに符号化する	yes
pgp_timeout	パスフレーズをキャッシュしている時間	300
pgp_verify_sig	署名を常に検証する	yes
pgp_create_traditional	旧形式のPGPメッセージを作成する	no
forward_decrypt	暗号化されたメッセージを転送時に復号化する	yes
pgp_sign_as	複数の鍵ペアを持っている場合にどの秘密鍵を使うかを指定する	""
pgp_long_ids	64ビットの鍵IDを使う	no
pgp_ignore_subkeys	サブキーを無視する	yes
pgp_sort_keys	PGP鍵の選択メニューで鍵をソートする項目を指定する	address
pgp_show_unusable	PGP鍵の選択メニューに、使用不可の鍵も表示する	yes
pgp_entry_format	PGP鍵の選択メニューでの鍵の表示形式	"%4n %t%f %4l/0x%k %-4a %2c %u"
pgp_autosign	作成したメッセージにPGP署名を常に付ける	no
pgp_autoencrypt	作成したメッセージを常に暗号化する	no
pgp_replyencrypt	暗号化されたメッセージに返信するとき自動的に暗号化する	no
pgp_replysign	署名されたメッセージに返信するとき自動的に暗号化する	no
pgp_replysignencrypted	暗号化されたメッセージに返信するとき自動的に署名する	no

りません。

●基本設定

PGP関連の設定変数は先に述べたラッパー変数だけでなく、表1に示したようなものがあります。この中で特に設定が必要であると思われるものおよび説明が必要であると思われるものについて説明します。

●pgp_strict_enc

PGP/MIMEの署名をしたメッセージが、配送経路上で「From_」行対策*7の「>」が挿入されたり、行末のスペースが除去されたり、8ビットコード

【実行例3】PGPを使うためのインストールオプション

```
$ make clean
$ ./configure --enable-pgp <その他必要なオプション>
$ make
# make install
```

【リスト4】符号化された文字列

```
--ew6BAiZeqk4r7MaW
Content-Type: text/plain; charset=iso-2022-jp
Content-Disposition: inline
Content-Transfer-Encoding: quoted-printable

=46rom hell.

--=20
TAKIZAWA Takashi (=1B$BBl_7=1B(B =1B$BN4;K=1B(B)

--ew6BAiZeqk4r7MaW
```

＊7　mbox形式のスプールでは、メッセージの区切りに「From foo@example.org」のような「From 」で始まる行を使用しているため、本文中に「From」で始まる行があると、行頭に「>」を挿入してメッセージの区切りとは区別するようになっています。これはメッセージが改ざんされているわけですから、署名の検証には失敗してしまいます。
＊8　Base64とQuoted-Printableは8ビットのデータを安全に転送するために、7ビットである表示可能なASCII文字列に符号化する規則です。Base64は、8ビット3文字を6ビット4文字に分割し、64個のASCII文字列（A～Z、a～z、0～9、+、-）に変換します。Quoted-Printableは、8ビット文字や制御文字などの文字1バイトを「=」に続けて16進数の2文字で符号化する方法です。

Column

Muttも歩けば棒に当たる

単なるMuttユーザーの川口です。以前はMutt用の非常にちゃちなパッチを作ったりもしていたのですが、最近は自分では使い方も落ち着いてしまったせいもあって、ほとんど滝澤さんにおまかせ状態です。

もともとMewを使っていたのですが（一時期MTAにqmailを使っていて）、「Maildir対応」というキーワードでMuttの存在を知りました。そこで「ターミナル中で動く」、「色がビシバシ」、「結構高機能／カスタマイザブル」という辺りでMuttに興味を持ちました。

Mewでも別に良かったのですが、日本語入力にkinput2を使っているため、「emacs -nw」な環境*9で色が表示できなかった（最近はターミナル中でも色が表示できるらしいのですが……）というような事情もあります（ちなみに、エディタには超軽量Emacsモドキのngを使っています）。

Mutt（＋日本語パッチ）の存在を知ってからも、しばらくはFreeBSDおよびDEC OSF/1（1996年頃）でうまく日本語が扱えるようにできなくて放り出していました。バージョンは0.7xくらいだったと思います。バージョン0.95.xくらいになったころ、再挑戦する気になり、mutt-jの皆さんに助けられながら、ついに使えるようになりました。それからは、主MUAとしてMuttを利用しています。

■mh_path機能の宣伝

せっかく記事を書くことになったので、Muttに自分で追加した機能をちょっと宣伝しておきます。mmh_path機能は、メールの保存形式にMHフォルダを使い、Mutt以外のメーラも使う人向けの機能です。

Muttは他の多くのMUAとは異なり、複数のメールフォルダ形式での読み書きをサポートしているため、それぞれの形式を判別する機能を持っています。

MHフォルダでは、オリジナルのMHが残す「.mh_sequences」というファイルの存在を判定に使うのですが、必ずしもMHフォルダを利用するMUAのすべてがこのファイルを作るわけではないので、あるフォルダをMuttで読もうとすると「mailboxじゃないよ」とか言われて、いちいち修正しなければならないので面倒です（それどころかせっかく「Muttを試してみよう」と思った矢先にそんなメッセージが表示されては、理由も分からず試すのをやめたくなってしまう人すらいるかもしれません）。

私はMew出身だったので、最初は.mew-cacheを判定に追加していたのですが（ありがち）、「ことはMew対策だけでは済みそうにない」と分かった時点で「mh_path」というものを突っ込んでみました。具体的には$mh_pathという変数を追加し、それを

```
set mh_path=~/Mail
```

のようにしてセットしました。すると「~/Mail/以下すべてのディレクトリはMHフォルダである」と見なすようにしました（ただしdirectory browserの都合で、サブディレクトリを抱えている親ディレクトリは判定条件からは外しています――サブディレクトリの方はもちろん対象です）。

主に使うメーラはMuttだけれど他のツールも使うとか、Muttでnews archiveを読みたいとか、リードオンリーのフォルダで自分ではtouch .mh_sequencesが実行できない場合などに使うと便利な機能です。

■MHフォルダ党

Muttとは直接には関係ありませんが、「MHフォルダ」の話を少々。Muttはmbox、MMDF、MH、Maildirなど、さまざまなフォルダ形式をサポートしているので、いろいろな流派の人が自分の好みのスタイルで自由に利用しています（この辺の雰囲気はまさに「Muttならでは」ですね）。

他の流派の人から、「MHフォルダを使うのはなぜか」と聞かれることも多いので、ここで簡単に「私がMHフォルダを使う理由」を書いてみましょう。

私はこれまで、MH→Mew→Muttという変遷を経てMUAを使ってきたわけですが、今でもときどきシェルからメールをいじるツールとして、MHやIMなどを使うことがあります。そのため、メールの保管にはMHフォルダを使っています。また、MHフォルダは他にも多くのMUAが対応しているので、Muttだけでなく、あっちのMUA、こっちのMUA、などいろいろ試すことができるのも便利ですね。

MHフォルダは、構成が非常にシンプル（あるディレクトリの下に1から番号順でメールが独立ファイルとして置かれる）なので、MUAを使わなくてもそれなりに扱うことができるのが便利です（と個人的に思い込んでいます）。

まあ、Maildirでも良いのですが（こちらも利用してはいます）、MHフォルダだとzshの「数値」の取り扱いと組み合わせると、MUAを使わずとも「ある範囲指定の操作」などができます。Tipsとかいうほどのものではないですが、例えば、

- {7..12}
  「7 8 9 10 11 12」に展開
- <7-12>
  「7～12」の数字のうち、存在するものにマッチ

といった類は便利ですね（こんなものを便利に感じるというのはMUAを使いこなしていないだけかもしれませんが）。ただ、Muttでは他の多くのMUAと異なり、（MUAにおける）メールの番号からそのメールのファイルを特定するのが簡単、という要素はなくなってしまっているのがちょっと残念です。

■MS Office対策

UNIX系統のOSを日常生活でも利用している人（特に会社員など）の多くの人は、おそらく理不尽な（？）「MS Office文書攻撃」を受けているのではないかと思います。基本は「そのようなものは避ける」といきたいところなのですが、そうも言っていられないこともあります。そこで、Microsoft Word／Excelについてはそれなりに読めるツールを1つご紹介しておきます（PowerPoint用のツールについてはよく知らないので割愛させていただきます）。

Wordのファイルは「wvWare」（[12]）というツールにlv、w3mを組み合わせて読むことができます。図の部分もある程度はデコードしてくれるようですが、**実行例6**に示すように、私はテキストを読むことにしか利用していません。そして、得てしてそれで十分な文書が9割だったりします。

lvをunicode変換に使っていることで分かるように、日本語も読めます（docファイルのバージョンにもよるのだと思いますが、そこまで調べてはいません）。

Excelのファイルは、xlHtml（[13]）というツールを使います。実行例7のようにするとファイルの中身を読むことができます。

どちらも「cat "$@"」としているのは、muttのページャでview attachmentからパイプで使うことができるようにするためです。xlHtmlの出力の方は、ファイルの内容によってはktermの横幅を広げたり、w3mに-colsオプションをつけて横幅を広げたりした方がよいかもしれません。

最近はPCが安くなったので、Windowsマシンを隣に用意してWordやExcelのファイルを読むという人も多いかもしれませんが、わざわざ画面を切り替えて見るまでもないような文書の場合は、こんな手でそれなりに読むことができます。　（川口銀河）

【実行例6】Wordのファイルを読むツールwvWareの使用例

```
#!/bin/sh
mkdir -p /tmp/wv
cat "$@" > /tmp/wv/cat.$$
wvHtml /tmp/wv/cat.$$ /tmp/wv/doc.$$
cat /tmp/wv/doc.$$ | lv -Iu8 -Oj | nkf -e -Z | w3m -T text/
```

【実行例7】Excelのファイルを読むツールxlHtmlの使用例

```
#!/bin/sh
cat "$@" > /tmp/xls.$$
xlHtml /tmp/xls.$$ | lv -Iu8 -Oj | nkf -e -Z | w3m -T text/html
```

が含まれるなどの理由でBase64とQuoted-Printable*8に符号化されたりして、署名の検証に失敗するケースがあります。

変数$pgp_strict_encが有効になっていると、例えば本文中に「From_」行があるとき、「F」は「=46」に符号化されたりして、行末スペースも「=20」となります。これによって、配送経路上での改変による署名の検証の失敗を防ぐことができます。この変数はデフォルトで有効になっています。

リスト4に、$pgp_strict_encが有効になっていることで、文字列がQuoted-Printableで符号化されている例を示します。ただし、日本語（ISO-2022-JP）を使う場合は、ESC文字（0x1B）が符号化されてしまい、相手側のメーラにとっては都合が悪くなることもあります。そのようなときには、

```
set pgp_strict_enc=no
```

として$pgp_strict_encの設定を無効にしてください。設定を無効にした場合には、「From_」行の扱いに注意してください。

*9　ターミナル内でEmacsを動作させている場合のこと。

●pgp_verify_sig

$pgp_verify_sigが「yes」の場合、Muttは PGP/MIMEの署名の検証を自動的に行おうとします。デフォルトでは「yes」になっています。

この設定が「yes」になっていると、署名したメッセージを表示するたびに検証を行うために、鍵リングにない公開鍵を鍵サーバへ探しにいきます。読み飛ばしたいメッセージがあっても、鍵を取得しようとして待たされるので、ちょっとストレスが溜まります。こうなると、必要なメッセージだけを検証したいと思うようになるので、検証するかどうかを訪ねるように「ask-yes」あるいは「ask-no」に変更します。

```
set pgp_verify_sig=ask-yes
```

●pgp_good_sign

Muttは、PGP署名の検証の成功をPGPコマンドの終了コードが「0」であるかどうかで判断しています。しかし実装によっては、検証が失敗しても終了コードを「0」にするものもあるため、成功したかどうかの判断はつけられません。このような場合には$pgp_good_signを使用します。

署名を検証したときのPGPコマンドの出力に$pgp_good_signに設定した文字列（例えば「Good signature」）があれば、署名の検証が成功したと判断することができるようになります。ロケールの設定によっては、出力のメッセージが日本語である場合があるので、このとき$pgp_good_signに記述する内容はリスト5のように日本語にします。

設定したにもかかわらず、判定がうまくいかないときは、ページャに表示されているPGPコマンドの出力結果を見れば、正しい検証結果を知ることができます。

【画面1】公開鍵サーバのページ

【リスト5】$pgp_good_signの設定例

```
set pgp_good_sign="(Good signature|正しい署名)"
```

【実行例4】公開鍵の取得方法（相手の公開鍵の鍵IDを指定する場合）

```
$ gpg --recv-keys 0x01234567
gpg: 鍵01234567をpgp.nic.ad.jpに要求 ...
gpg: 鍵01234567: 公開鍵を読み込みました
gpg:           処理数の合計: 1
gpg:                 読込み: 1
```

【実行例5】公開鍵の取得方法（公開鍵ファイルをインポートする場合）

```
$ gpg --import pubkeyfile
gpg: 鍵01234567: 公開鍵を読み込みました
gpg:           処理数の合計: 1
gpg:                 読込み: 1
```

おまけのTips　アドレスの入力を横着するには

新規話題のメッセージにもかかわらず、アドレスの入力を横着して他のメッセージの返信の形式で出す人をメーリングリストなどでときどき見かけます。スレッド表示でメッセージを読んでいると、スレッドツリーの途中で新規話題が始まるので、あまり気持の良いものではありません。

横着の仕方にもいろいろな技があるもので、Muttの場合は、他のメッセージへの返信の形式でメッセージを起こしても、編集時にヘッダの「In-Reply-To:」フィールドの行を削除すれば、「In-Reply-To:」、「References:」フィールドのないメッセージが作成されて、他のメッセージにぶら下がるようなことがなくなります。ただし、返信元のメッセージには返信フラグが付けられます。

まあ、エイリアスへの登録は簡単にできるので、できるだけエイリアスを使ってほしいところなのですが……。

（滝澤隆史）

●pgp_retainable_sigs

$pgp_retainable_sigsは署名をしてから暗号化する場合の動作を指定します。「no」の場合は署名と暗号化を同時に行ういます。「yes」の場合はPGP/MIMEの署名メッセージを作成し、そのマルチパートのメッセージ全体に対して暗号化を行います。

■使用例

以上で必要な設定がすべて終わりました。それでは実際に使ってみましょう。

●公開鍵の取得方法

Muttはは PGP署名の検証時には相手の公開鍵の鍵IDが分かるため、公開鍵サーバから自動的に取得します。しかし、暗号化を行うときには相手の公開鍵の鍵IDが分からないため、公開鍵の鍵リングにない場合は使うことができません。そのため、あらかじめ公開鍵を取得しておく必要があります。

相手の公開鍵の鍵IDが分かっている場合は、単に実行例4のようなコマンドを入力すれば良いだけです。

鍵IDが分からない場合は、公開鍵サーバのページで検索します。日本国内の場合はJPNICのページ（[10]、画面1）で検索すると良いでしょう。鍵IDを見つけたら、実行例4と同じようにして取り込むことができます。

公開鍵サーバに登録せずに、自身のWebページで公開鍵のファイルを公開している人がいます。このときは、ダウンロードした公開鍵のファイルを実行例5のようにしてインポートします。

●署名

まず、通常と同じようにメッセージを作成してください。Compose画面の上半分の情報欄のPGPの項目が「暗号化無し」になっています。ここで「p」(pgp-menu)を入力すると、PGPのメニューが表示され、PGPの処理をどうするか尋ねられます（画面2）。署名を行う「s」を入力すると情報欄のPGPの項目は「署名」になります。続いて「y」と入力すると、秘密鍵のパスフレーズの入力が求められます。パスフレーズを入力するとメッセージが送信されます。

●暗号化

まず、署名の時と同じように、メッセージを作成してからCompose画面でPGPのメニューを表示します。暗号化を行う「e」を入力すると情報欄のPGPの項目が「暗号化」になります。続いて「y」と入力すると、宛先のメールアドレスに一致する公開鍵の一覧が表示されます(画

【画面2】署名の設定

【画面4】署名の検証確認

【画面6】パスフレーズの入力

【画面3】公開鍵の選択

【画面5】署名の検証成功

【画面7】復号化されたメッセージ

面3)。ここで、使用する公開鍵を選ぶとメッセージが送信されます。

●署名の検証

PGP署名されたメッセージにはフラグ「s」が付けられます。$pgp_verify_sigが「ask-yes」になっている場合には、署名の検証を行うかどうかを尋ねるメッセージが表示されます(画面4)。ここで「y」と入力すると、画面5のように表示されます。

旧形式のPGP署名のメッセージの場合、Muttは自動認識しません。署名の検証を行いたい場合は「Esc P」を入力します。

●復号化

暗号化されたメッセージにはフラグ「P」が付きます。このメッセージを表示しようとすると、秘密鍵のパスフレーズの入力が求められます(図6)。入力すると復号化され、図7のようにメッセージが表示されます。最下行に「PGP署名の検証失敗」とのメッセージが出ていますが、これは無視します。

旧形式の暗号化メッセージの場合はMuttは自動認識しません。復号化したい場合は「Esc P」を入力します。

■最後に

メッセージの暗号化／電子署名を扱うもう1つの規格であるS/MIMEをMuttで使えるようにするパッチがあります([11])。個人で使うことはあまりないと思いますが、業務でS/MIMEが必要な方は試しに使ってみてはどうでしょうか。

Resource

[1] 書籍「PGP 暗号メールと電子署名」
http://www.oreilly.co.jp/BOOK/pgp/
Simson Garfinkel著／山本和彦監訳／株式会社ユニテック訳／オライリー・ジャパン発行／5050円／ISBN4-900-90002-8

[2] 書籍「E-Mail セキュリティ」
http://www.ohmsha.co.jp/data/books/contents/4-274-06117-5.htm
Bruce Schneier著／力武健次監訳／道下宣博訳／オーム社発行／3398円／ISBN4-274-06117-5

[3] (仮) 日本の公式 PGP ホームページ
http://pgp.iijlab.net/

[4] PGP User's Manual for Windows
http://www.cla-ri.net/pgp/

[5] 日本ネットワークアソシエイツ
http://www.nai.com/japan/

[6] The GNU Privacy Guard
http://www.gnupg.org/

[7] Gnu PG 配布元
ftp://ftp.gnupg.org/GnuPG/

[8] Ring Server の gnupg ディレクトリ
ftp://ftp.ring.gr.jp/pub/net/gnupg/

[9] GnuPG + IDEA for Windows ＊2001年8月10日時点でアクセスできませんでした。
http://www.nullify.org/

[10] PGP PUBLIC KEYSERVER ／ PGP.NIC.AD.JP
http://pgp.nic.ad.jp/jindex.html

[11] S/MIME for Mutt
http://elmy.myip.org/mutt/smime.html

[12] Word のファイルを読むツール「wvWare」のページ
http://wvWare.sourceforge.net/

[13] Excel のファイルを読むツール「xlHtml」のページ
http://www.xlhtml.org/