さらに進んだ インターネット・セキュリティ

サーバー構築/運用術

第11回 メール(中編)

迷惑メールが社会問題になるほど，メールは人々の生活に密着するものになりました。前回はMTA間のメール配送について説明しましたが，今回はその前段階である，ユーザーがメールを送信してからMTAに届くまでについて，説明します。 **(ケイ・ラボラトリー 仙石 浩明)**

明けましておめでとうございます。21世紀最初の年が終わりましたが，読者の皆様にとって昨年はどのような年でしたでしょうか。残念ながら社会全体としてみれば不況やテロなど暗い話題の多い年でしたが，2001年1月にJavaを搭載した携帯電話(以下，Java搭載携帯電話)が発売され，一気に普及した年でもあります。わずか1年弱の間に1千万人以上の人が携帯電話上でJavaアプリを使い始めたという，Javaの歴史にとっても，携帯電話の歴史にとっても，歴史に残る年だったと言えるでしょう。

そして11月末には横浜でJavaOne*1が開催されました。Java搭載携帯電話によって一気にJava普及率ナンバー・ワンに踊り出た日本が，米国以外での初のJavaOne開催地として選ばれたのは当然の結果だったのでしょう。日本で開催されたということで，JavaOne会場に足を運ばれた方も多かったのではないでしょうか*2。

私自身は，開催の直前までJavaOne公式アプリ*3のデバッグに追われ，開催初日の晩は BOFのスピーカ*4を務め，連日深夜まで飲み会があったので大変だったのですが，無事終わってほっとしています。

メール送信

メール配送の概略を**図1**に示しました。メールを配送しようとするメール配送エージェント(Message Transfer Agent=MTA)は，まずネーム・サーバーへ問い合わせることによってメールのあて先アドレスに対応するメール交換局(Mail eXchanger=MX)のホスト名を得ます。MXでは，インターネット上のその他のホストからのメールを受信するためのMTAが動作しています。

例えば，これから配送しようとするメールのあて先がsengoku@gcd.orgである場合，ネーム・サーバーを使ってgcd.orgのMXレコードを検索し，あて先のMXであるmx.gcd.orgを得ます。

MTAが行っている問い合わせと同等のことをnslookupコマンドを使って行ってみた例を**図2**に示します。「mail exchanger=mx.gcd.org」と表示されているので，gcd.org に対応するMXが「mx.gcd.org」であることが分かります。

次にMTAは，あて先のMX上のMTAへSMTP*5接続します。つまりメ

＊1 Java開発者を対象とした世界最大のコンファレンスです。毎年，米国のサンフランシスコで開催されています。
＊2 さすがに約10万円の入場料をポンと出せる人は多くはないと思いますが，コンファレンス等を聴講できないパビリオンのみの入場招待券が多数配られたので，多くの人が参加したようです。
＊3 JavaOne公式アプリは，「JavaOneテクノロジーパートナー」であるケイ・ラボラトリーが開発・提供しました。初日の基調講演では，米Sun MicrosystemsのJohn Gage氏による公式アプリのデモンストレーションが行われました。
＊4 「ケータイ Java プログラミング」と題して，携帯電話で動くJavaアプリを開発する上での留意点を中心に解説を行いました。
＊5 SMTP (Simple Mail Transfer Protocol)について，詳しくはRFC 2821を参照してください。

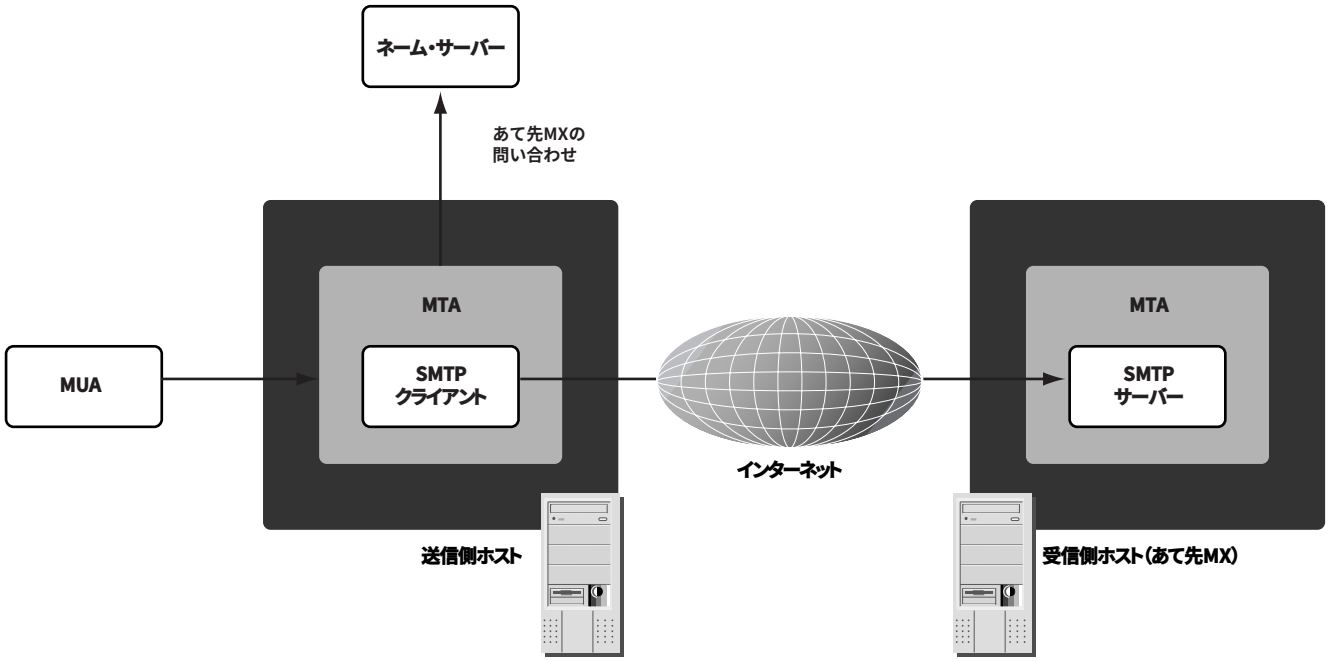

図1　メール配信

ールを送信する側のMTAがSMTPクライアントの役割を，メールを受信する側のMTAがSMTPクライアントの役割を果たします。

さて，では送信側のMTAは，送信すべきメールをどのように得るのでしょうか。ユーザーがメールを読み書きするためのソフトウエアをMUA（Message User Agent）と呼びます。MUAは何らかの方法でユーザーが書いたメールをMTAへ伝えなければなりません。

MTA直接起動

MUAがMTAへメールを伝える最も単純な方法が，MUAがMTAのプログラムを直接起動する方法です。ただしMUAがMTAと同じホスト上で動作している必要があります。例えばMUAが**図3**のようなメールmail.txtを送信しようとしたとき，MTAがsendmailならば**図4**のように起動することになりますし，MTAがqmailならば**図5**のように起動する*6ことになります。

MTAをリモート起動

MUAがMTAと異なるホスト上で動作していても，リモート起動することができれば，ほとんど同様の方法でMTAを起動できます。リモートからMTAを起動する場合，MTAの種類を気にすること無く起動できると便利ですから，MTAを起動するだけの単機能のプログラムrmailを使います。rmailの起動は，**図6**に示すように，引数にあて先アドレスを与え，メール本体は標準入力から与えます。

すると，sendmailを使っているホストでは，rmailを実行するとsendmailプログラムが内部的に呼び出され，qmailを使っているホストではqmail-injectプログラムが内部的に呼び出されると言うわけです。MTA が動いているホストの名前をremoteとすると，例えばsshを使って**図7**のように，MTAを起動できます。

＊6　MTAのすべての機能が1つのプログラムに含まれているsendmailと異なり，qmailは機能ごとに小さいプログラムに分かれています。MUAから起動してメールをMTAへ伝えるためのプログラムがここで例示したqmail-injectです。

また，UUCPを使ってリモート起動できるホストであれば，**図8**のようにMTAを起動できます。実際，UUCPによるメール配送では，MTAが隣接ホストのMTAを起動するために，この方法を用いています。

SMTP接続を行うMUA

1990年ごろから，MacintoshやWindowsなどのPC用OSにもTCP/IPが実装されるようになり，それまでUNIXマシンの独擅場だったLANにPCが接続されるようになってきました。始めはPCからtelnetなどを使ってUNIXマシンへリモート・ログインしてメールの読み書きをしていたユーザーたちは，すぐに使い慣れたPC上でメールを読み書きしたいと思うようになります。

当時，PCからUNIXマシン上のプログラムをリモート起動する方法は一般的ではなかったため，PC上のMUA*7のほとんどすべては，UNIXマシン上のMTAへメールを伝えるためにSMTPを使う方法を採用しました。つまり当時のMTAは，自分あてでないメールが届いたら，あて先のMXへ中継するのが普通でした*8から，PCからSMTP経由でMTAにメールを送っても，正しくあて先

```
azabu:/home/sengoku % nslookup ↵
Default Server:  azabu
Address:  0.0.0.0

> set type=mx ↵
> gcd.org. ↵
Server:  azabu
Address:  0.0.0.0

gcd.org preference = 10, mail exchanger = mx.gcd.org
gcd.org nameserver = ns.gcd.org
gcd.org nameserver = brother.daio.net
gcd.org nameserver = ns-tk012.ocn.ad.jp
mx.gcd.org      internet address = 210.145.125.162
ns.gcd.org      internet address = 210.145.125.162
brother.daio.net        internet address = 210.167.164.35
>
```

図2 メールのあて先からMXを検索 ↵このマークで改行
gcd.orgに対応するMXが，mx.gcd.orgであることが分かる。

```
From: sengoku@klab.org
To: sengoku@gcd.org
Subject: test

This is a test mail, sorry.
```

図3 テスト用のメールmail.txt

```
sendmail -oi sengoku@gcd.org < mail.txt
```

図4 MTAがsendmailの場合の起動方法

```
qmail-inject sengoku@gcd.org < mail.txt
```

図5 MTAがqmailの場合の起動方法

```
rmail sengoku@gcd.org < mail.txt
```

図6 rmailの起動方法

```
ssh remote "rmail sengoku@gcd.org" < mail.txt
```

図7 sshを使ったリモートのMTA起動

```
uux - remote!rmail sengoku@gcd.org < mail.txt
```

図8 UUCPを使ったリモートのMTA起動

*7 PC上のMUAとしてはEudoraが有名です。1988年に発表された後，Webブラウザ付属のMUAに取って代わられるまで，最も人気のあるMUAの1つでした。
*8 本連載2002年1月号(前号)を参照。

へ届いたのです(**図9**参照)。

そして, SMTP接続を行うMUAはPC用だけでなくUNIX用のMUAでも主流になり, ついには MUAがMTAと同じホスト上で動作している場合でさえ, MTAを直接起動するのではなく, localhostへSMTP接続してメールを送るMUAが主流になりました。

認証機能付きMTA

1990年代後半に入って不正中継が問題となったため, MTAでは原則中継を禁止し, サイト内部のIPアドレスからのSMTP接続のみ中継を許可するようになりました。

ところがこれではサイト外部のPC, 例えばプロバイダにダイヤルアップしてインターネットへ接続したモバイルPCから普段使っているMTAが使えなくなります。

もちろん, プロバイダにダイヤルアップしているときはプロバイダが提供するMTAを使い, 職場のLANなどに接続しているときは職場のMTAを使うように, そのつどMUAの設定を変更すれば解決する問題ではあるのですが, 次の理由などにより, 常に職場のMTAを使えると便利でしょう。

・プロバイダのMTAを使いたくない

負荷が高すぎて, メールの送信に時間がかかりすぎるなど, プロバイダのMTAの品質に問題がある場合など。

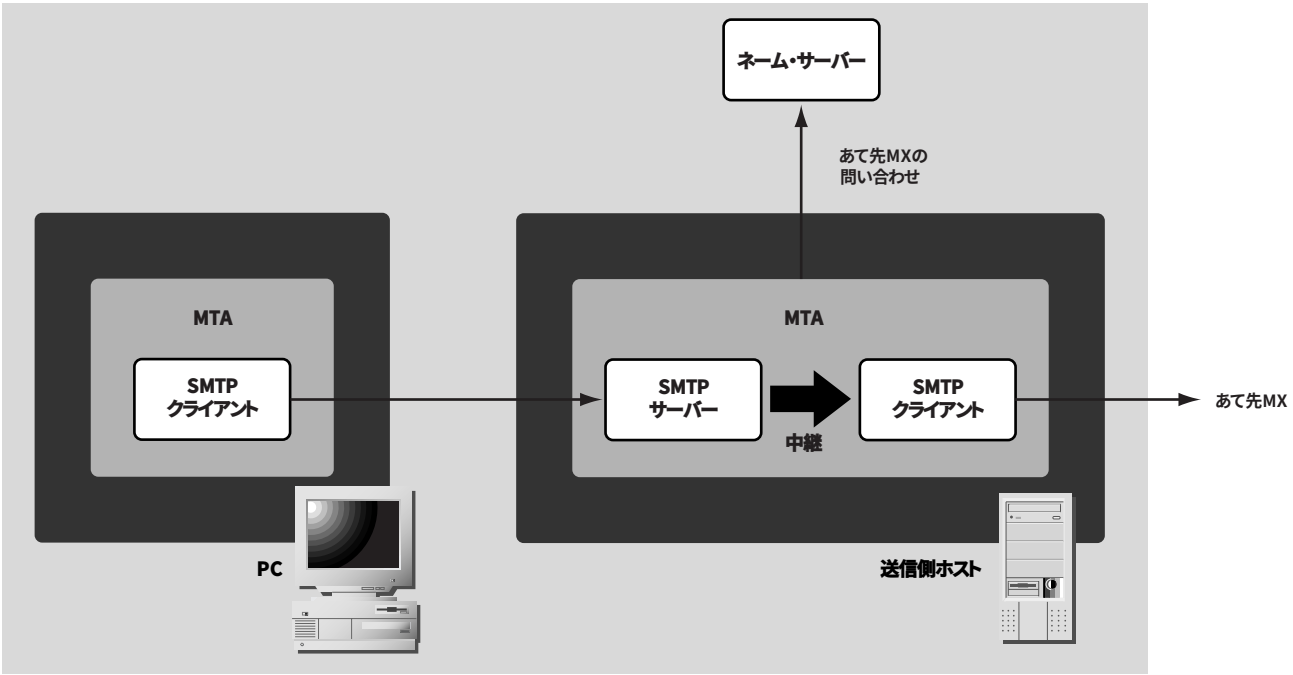

図9　MUAからSMTPでMTAへ接続

・プロバイダからMTAが提供されていない

特に海外ローミング・サービスを利用する場合など，常にMTAが提供されるとは限りません。また，日常的に利用しているわけではないプロバイダの場合だと，どのMTAを利用すればいいのかをいちいち調べるのは面倒でしょう。

・職場のMTAがSMTP接続の暗号化をサポートしている

職場のアドレスあてのメールを送る場合，暗号化したSMTPで職場のMTAへ接続すれば，機密情報も送ることができます。もちろん職場以外のアドレスあてに送る場合は，MTAからあて先MXまでの通信路は暗号化されませんが，あて先ごとに利用するMTAを切り替えるのは面倒です。

そこで，MTAを利用する際に，ユーザーが自身を認証してもらう仕掛け「SMTP 前 POP」(POP before SMTP)と「SMTP認証」が考案されました。認証に成功した場合，MTAは任意のあて先への中継を許可し，認証に失敗した場合あるいはMUAから認証のリクエストが無かった場合は，自ドメインあて以外のメールの受信を拒否することになります。

SMTP前POP (POP before SMTP)

SMTP接続の前に，いったんユーザーにPOP(Post Office Protocol, 次号で解説予定)でアクセスさせて，POPにおける認証に成功した場合は，そのユーザーからのSMTP接続に限り中継を許可する方式です。POP接続とSMTP接続が同じIPアドレスから行われていて，かつ2つの接続が一定時間(例えば5分)以内に行われた場合に限り，同じユーザーからのアクセスであると判断します。

メールの受信操作を行ってから送信操作を行えば，どんなMUAでも利用可能なので，後述するSMTP 認証機能をサポートするMUAが普及するまでの「つなぎ」と言えます。

SMTP認証

SMTPに認証機能を拡張する方式です*9。SMTPクライアントとサーバー間の通信を簡単にまとめると次のようになります。

(1) サーバーが応答コード220を送信

(2) クライアントが「EHLO ホスト名」コマンドを送信してSMTP拡張に対応していることをサーバーへ示す。

(3) サーバーが応答コード250に続けてサポートしている拡張コマンドの一覧を送信。

(4) クライアントが「AUTH 認証方法」コマンドを送信して認証方法を指定する。

(5) サーバーが応答コード334に続けて「チャレンジ」を送信。

(6) クライアントが「レスポンス」を送信。

(7) サーバーが続けて「チャレンジ」を送信する場合は(5)へ戻る。
認証成功の場合は，応答コード235を返す。
認証失敗の場合は，応答コード535を返す。

なお，SMTP拡張*10に対応していないクライアントの場合は，上記(2)にお

*9 詳しくはRFC 2554を参照。
*10 詳しくはRFC 2821を参照。

```
 1  220 asao.gcd.org ESMTP
 2  EHLO takatsu.gcd.org
 3  250-asao.gcd.org
 4  250-AUTH LOGIN CRAM-MD5 PLAIN
 5  250-PIPELINING
 6  250 8BITMIME
 7  AUTH LOGIN
 8  334 VXNlcm5hbWU6
 9  c2VuZ29rdQ==
10  334 UGFzc3dvcmQ6
11  c2VjcmV0IHBocmFzZQ==
12  235 go ahead
13  MAIL FROM:<sengoku@gcd.org>  BODY=8BITMIME
14  250 ok
15  RCPT TO:<sengoku@klab.org>
16  250 ok
```

図10　LOGIN 認証

```
 1  220 asao.gcd.org ESMTP
 2  EHLO takatsu.gcd.org
 3  250-asao.gcd.org
 4  250-AUTH LOGIN CRAM-MD5 PLAIN
 5  250-PIPELINING
 6  250 8BITMIME
 7  AUTH CRAM-MD5
 8  334 PDQ4NzQuMTAwNzI3Njc3N0Bhc2FvLmdjZC5vcmc+
 9  c2VuZ29rdSA1NjJlYWVhY2RlNmM0MjMzZDYzYzY2YjFkMjY4ZjU4Ng==
10  235 go ahead
11  MAIL FROM:<sengoku@gcd.org>  BODY=8BITMIME
12  250 ok
13  RCPT TO:<sengoku@klab.org>
14  250 ok
```

図11　CRAM-MD5 認証

```
レスポンス = H(パスワード XOR opad, H(パスワード XOR ipad, チャレンジ))

ipad =  (0x36を64回繰り返した文字列)
opad =  (0x5cを64回繰り返した文字列)
```

図12 かぎ付きメッセージ・ダイジェスト

いて「HELO ホスト名」コマンドを送信します。サーバーがSMTP拡張に対応していない場合は,「EHLO ホスト名」コマンドに対して応答コード500を返すので, クライアントは改めて「HELO ホスト名」コマンドを送信する必要があります。

SMTP認証の認証方法には「LOGIN」「CRAM-MD5」「PLAIN」などの方法があり, サーバーがどの認証方法に対応しているかは, (3)でサーバーが送信する応答で知ることができます。その中から, クライアントは認証方法を1つ選んで(4)で指定します。(5)と(6)が実際の認証になります。

サーバーが「チャレンジ」をクライアントへ送り, それにクライアントが「レスポンス」で応えることで認証が行われます。必要ならばサーバーは何度も「チャレンジ」を繰り返し, クライアントに「レスポンス」を要求できます。実際に送信される「チャレンジ」および「レスポンス」は, BASE64でエンコードした文字列です。

図10にLOGIN認証の実例を示します。行頭が3けたの数字すなわち応答コードで始まっている行がサーバーが送信した行で, それ以外がクライアントが送信した行です。ここでは, ユーザー

「sengoku」のパスワードが「secret phrase」であるとしています。

4行目が，サーバーがサポートしている認証方法の提示です。このSMTPサーバーでは，「LOGIN」「CRAM-MD5」「PLAIN」の3通りの認証方法が可能であることが分かります。これに対し，クライアントが7行目で認証方法として「LOGIN」を指定しています。

8行目が，サーバーが送信したチャレンジで，「VXNlcm5hbWU6」をBASE64でデコードすると「Username:」になります。つまり認証するユーザー名の要求です。これに応えて，クライアントが送信したレスポンスが9行目です。ユーザー名「sengoku」をBASE64でエンコードした「c2VuZ29rdQ==」を送信しています。

10行目は，サーバーが続いて送信したチャレンジで，「UGFzc3dvcmQ6」をBASE64でデコードすると「Password:」になります。つまりユーザーのパスワードの要求です。これに応えて，クライアントが送信したレスポンスが11行目です。パスワード「secret phrase」をBASE64でエンコードした「c2VjcmV0IHBocmFzZQ==」を送信しています。

すると12行目にあるように，応答コード235がサーバーから返ってきて認証が成功したことが分かります。

LOGIN認証は，ユーザー名およびパスワードをBASE64でエンコードしただけで送信しますから，通信路の暗号化*11が不可欠になります。そもそも，せっかくサーバーがチャレンジを提示してクライアントがレスポンスを返すわく組みであるにもかかわらず，チャレンジが毎回同じ「Username:」と「Password:」では意味がありません。

サーバーが送信するチャレンジが毎回変化し，それに応じてクライアント側が毎回異なるレスポンスを返さなければならない認証方式としてCRAM-MD5があります。CRAM-MD5はIMAP（Internet Message Access Protocol，次号で解説予定）あるいはPOPにおける認証方法として，RFC 2195で規定されています。この認証方法は，サーバーとクライアントが共通のかぎ（つまりパスワード）を持っている場合に利用可能で，まずサーバーが「<4874.1007276777@asao.gcd.org>」などのタイムスタンプを含む文字列をチャレンジとして送信し，それに対してクライアントがチャレンジとパスワードからレスポンスを算出して送り返す方法です。サーバーにおいても，同様の方法で算出した値が，クライアントから送られて来たレスポンスに一致すれば認証成功となります。

図11にCRAM-MD5認証の実例を示します。7行目でクライアントが認証方法として「CRAM-MD5」を指定しています。8行目が，サーバーが送信したチャレンジで，「PDQ4NzQuMTAwNzI3Njc3N0Bhc2FvLmdjZC5vcmc+」をBASE64でデコードすると「<4874.1007276777@asao.gcd.org>」になります。ここで「1007276777」はタイムスタンプで，1970年1月1日0:00GMTからの秒数を表わしています。

このチャレンジと，パスワード「secret phrase」からクライアントが送信すべきレスポンスを算出する方法は，かぎ付きメッセージ・ダイジェストと呼ばれている方法で，RFC 2104 で規定されています。算出式を**図12**に示します。

まずパスワードとipad（0x36を64回繰り返した文字列）を1文字づつXORを計算した64バイトのデータ列を作成します。例えば1文字目はパスワードが「s」つまり0x73，ipadが0x36ですから，0x73と0x36のXORを計算して0x45になります。同様の計算の繰り返しで得られる

＊11　2001年2月号「実践で学ぶ，一歩進んだサーバー構築・運用術」で解説したsshを使って暗号化する方法や，2000年9月号の同連載で解説したstoneを使ってSSLで暗号化する方法などがあります。

データ列を16進数で表現すると**図13**のようになります。

同様に，パスワードとopad（0x5cを64回繰り返した文字列）を1文字づつXORを計算した64バイトのデータ列を16進数で表現すると**図14**のようになります。

次に，図13の文字列にチャレンジ「<4874.1007276777@asao.gcd.org>」をつなげて，メッセージ・ダイジェスト5（MD5, 詳しくはRFC 1321を参照してください）と呼ばれるハッシュ関数に入力して，その値であるハッシュ値を計算すると，**図15**に示す16バイトのデータ列が得られます。

そして，図14のデータ列と図15のデータ列をつなげて，MD5ハッシュ関数に入力すると，**図16**に示す16バイトのデータ列が得られます。これを16進数で表現し，その前にユーザー名と空白文字をつなげた文字列，「sengoku 562eaeacde6c4233d63c66b1d268f586」がレスポンスとなります。

図11の9行目で，このレスポンスをBASE64でエンコードした「c2VuZ29rdSA1NjJlYWVhY2RlNmM0MjMzZDYzYzY2YjFkMjY4ZjU4Ng==」を送信しています。

SMTP認証でよく用いられる認証方式としては他にPLAINがあります。**図17**にPLAIN認証の実例を示します。PLAINは前述した2つの認証方式と異なり，AUTHコマンドと一緒に，パラメータとして認証情報を与えてしまうため，サーバーはチャレンジを送信しません。認証情報のフォーマットはRFC 2595で規定されています。

7行目で，クライアントが「AUTH PLAIN」に続けて認証情報を送ってい

```
45 53 55 44 53 42 16 46 5e 44 57 45 53 36 36 36
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36
```

図13 （パスワード XOR ipad）の計算結果

```
2f 39 3f 2e 39 28 7c 2c 34 2e 3d 2f 39 5c 5c 5c
5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c
5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c
5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c 5c
```

図14 （パスワード XOR opad）の計算結果

```
c4 80 32 89 3f 75 6c f7 cc 7a 67 1b cf de 08 dd
```

図15 H（パスワード XOR ipad, チャレンジ）の計算結果

```
56 2e ae ac de 6c 42 33 d6 3c 66 b1 d2 68 f5 86
```

図16 H（パスワード XOR opad, H（パスワード XOR ipad, チャレンジ））の計算結果

```
 1  220 asao.gcd.org ESMTP
 2  EHLO takatsu.gcd.org
 3  250-asao.gcd.org
 4  250-AUTH LOGIN CRAM-MD5 PLAIN
 5  250-PIPELINING
 6  250 8BITMIME
 7  AUTH PLAIN AHNlbmdva3UAc2VjcmV0IHBocmFzZQ==
 8  235 go ahead
 9  MAIL FROM:<sengoku@gcd.org>  BODY=8BITMIME
10  250 ok
11  RCPT TO:<sengoku@klab.org>
12  250 ok
```

図17 PLAIN 認証

ます。この認証情報は「0x00」「ユーザー名」「0x00」「パスワード」をつなげたデータ列をBASE64でエンコードした文字列となっています。

あて先MTAへ直接SMTP接続を行うMUA

MUAからMTAへメールを伝える方法を各種説明しましたが，そもそもMTAはさほど複雑な処理をしているわけではありません。ネーム・サーバーに対して，図2に示したような，MXの問い合わせを行い，得られたMXへSMTP接続を行うだけです。しかも，あて先MXのMTAへSMTP接続するのも，自サイトあるいはプロバイダなどが提供するMTAへSMTP接続するのも，何ら変わるところはありません。

従ってMUAにネーム・サーバーへMXを問い合わせる機能を追加するだけで，MUAから直接あて先MXのMTAへ接続してメールを送りつけることができるようになります。このようなMUAではMTAを必要としませんから，出先などでいろんなプロバイダにダイヤルアップする必要がある場合は便利です。

半面，このようなMUAからメールを送りつけられた側は，送信者の身元を調べようと思っても，手がかりは送信元ホスト，すなわちMUAが動作しているPCのIPアドレスだけです。ダイヤルアップ接続しているPCであれば，次の瞬間にはダイヤルアップを切断してインターネットから切り離されているかも知れません。

もちろん，メールを送りつけられた正確な時刻を記録していれば，プロバイダ側のダイヤルアップ接続の記録と突き合わせることにより，その時間にそのIPアドレスを割り当てられていたユーザーを調べることは不可能ではありません。しかし，よほどの理由がない限り，プロバイダはそこまでは調べてくれません。

送信者が自身の身元を明らかにするような事柄をメールに書かない限り，送信者を特定することはほぼ不可能であると言っても過言ではないでしょう。言い換えれば，送信者が同じプロバイダを利用している他のユーザーになりすまそうと思えば，ほぼ完ぺきにその目的が達成できるということです。

このようなMUAは迷惑メールの送信者にとって理想のツールと言えますし，実際にそのようなツールが出回っているようです。この種の迷惑メールを防ぐには，MTAにおいて，ダイヤルアップ接続しているPCからのSMTP接続を拒否すればよいでしょう。ダイヤルアップ用のIPアドレスのデータベースとしては，MAPS Dial-up User Listが有名です*12。

例えば接続元ホストのIPアドレス「a.b.c.d」の場合，DNSで「d.c.b.a.dialups.mail-abuse.org」を検索し，もしTXTレコードが登録されていたら，そのIPアドレスがデータベースに登録されていることを意味しますから，メールの受け取りを拒否します。

このような方法で，ダイヤルアップ用のIPアドレスから送信されたメールの受け取りを拒否するサイトもあるようですから，ダイヤルアップ接続しているPCからメールを送る際は，正規のMTAを経由して送る方が無難と言えるでしょう。

プロバイダによっては，外部のホストへのSMTP接続をフィルタリング*13しているところもあるようです。つまりユーザーはプロバイダが提供する正規のMTAを使うことを強制されます。そして，このMTAで中継するメールのヘッダーに，ユーザーのIDなどの情報を挿入するようにすれば，なりすましメールを防ぐことが可能になるでしょう。

＊12　残念ながらMAPSは2001年8月1日以降，MAPSと契約した人しか利用できなくなってしまいました。
＊13　迷惑メール送信の防止という観点では極めて有効な方法なのですが，ユーザーの立場から考えるとあまり利用したくないプロバイダになってしまいそうです。