インターフェイスの街角 (43) — 明るい認証システム

増井俊之

不便な認証システム

Web 上の情報サービスサイトや通信販売サイトでは、ユーザーを特定するためにパスワードによる個人認証がよく用いられています。パスワードによる認証システムは、実装は簡単ですが、ユーザーにとってはひどく使いづらいものが多いようです。自分の誕生日や親戚の名前など、ユーザーが簡単に暗記できるような文字列は、他人にも簡単に見破られるおそれがあります。一方、他人には推測しづらい文字列は、一般にユーザー本人にとっても憶えにくいため、紙に書き留めたりファイルとして保存しておかなければならなくなり、パスワードとしての意味をなさなくなってしまいます。

また、システムの都合でサービスごとにユーザー ID を割り当てられたり、制約のあるパスワードを強制されたりすることがあるのも困りものです。結果として、システムごとに異なるパスワードを用意したり、ID とパスワードを組にして憶えておかなければならなくなってしまいます。

ID やパスワードを完璧に暗記していたとしても、サービスを受けるたびに入力するのはけっこう面倒ですし、そもそもパスワードを盗まれたら一巻の終りです。パスワードを使わずに、手軽で安全な認証をおこなうことはできないものでしょうか。

認証が変える世界

Web のサービスにかぎらず、簡単で安全な認証が普及すれば世の中は大きく変わるはずです。

たとえば、電車に乗るときは切符を買うのが普通ですが、誰がどこで乗ったのかを簡単に認証する方法があれば、わざわざ切符という〝モノ〟を使う必要はありません。自動車を運転するときは、免許証の携帯が義務づけられています。しかし、これも免許の保持者であることを証明する方法があれば、つねに持ち歩く必要はないでしょう。完全な個人認証が実現すれば、現金はもとよりクレジットカードすら持ち歩かずにすみ、家や車の鍵も不要になる可能性があります。このように考えると、世の中の多くのカードや証明書などは、認証技術によって不要になるかもしれません。

現在のところ完璧な個人認証の実現は容易ではありませんが、ある程度は指紋や虹彩、声紋などによる認証も可能になりつつあります。将来的には、DNA などを用いた、さらに正確な認証ができるようになるかもしれません。ここまでくるとプライバシーや法的な問題なども出てくるでしょうが、あらゆるカードや証明書が不要になるのなら、そのほうがメリットが大きいようにも思えます。

〝明るい〟認証

とはいえ、指紋や DNA を認証に使うのはなんとなく気持ちが悪いものです。それに、システムを騙して他人になりすますことが絶対に不可能ともいえません。一方で、個人を特定できる装置を持ち歩く方法もいろいろと考案されています。しかし、単純な装置にすると盗まれた場合にはお手上げです。かといって、パスワードと組み合わせた装置にすると、カードを携帯して暗証番号を入力するのと大差なくなってしまいます。

どうも、認証というと明るい話は少なくて、悪事から身を守る方法とか、そのために多少の不便は我慢しようといった話が多いようです。簡単で楽しく利用でき、プライバシーの保護も完全かつ安全であるような〝明るい〟認証シ

ステムが欲しいところです。

脳内情報を用いた認証

個人を特定するには、脳のなかの情報を使うのがもっとも有効でしょう。同じ遺伝子をもつ一卵性双生児でも脳の中身は異なっているはずですし、脳には十分な量の情報が格納されています。中身を壊さずに覗いたりコピーしたりすることは、すくなくとも当分は不可能ですから、認証にはうってつけです。

パスワードの利用も脳のなかの情報を用いた認証手法の1つです。ただし、人間にとって難しい頭の使い方をしなければならない点が問題なのでしょう。他人に分からないようにするために、一般的な知識や推論可能な情報ではなく、意味のない文字列を使用するという方式をとっているため、ユーザー本人にとっても使いにくくなっているわけです。それなら、他人の脳では処理が難しいけれども、自分の脳ではすぐに処理ができるような情報を使えばよいはずです。

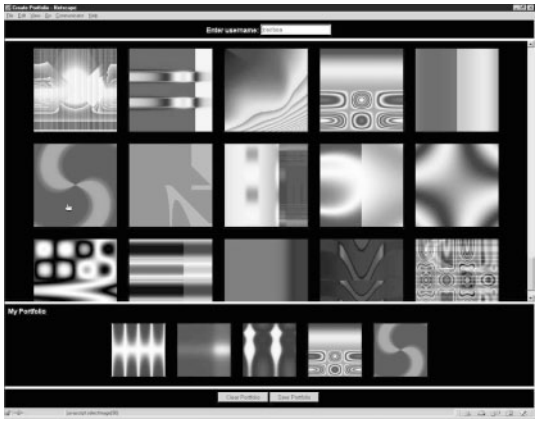
画像認識能力を利用した認証

パスワードの場合は文字列を正確に憶えておかなければなりませんが、文字列の代わりに画像を、入力操作の代わりに選択操作を使い、認証作業を楽にしようという考え方があります。

カリフォルニア大学バークレイ校の Adrian Perrig 氏と Rachna Dhamija 氏は、たくさんの画像のなかから、ユーザーが事前に好きな画像を選んでおくことによって認証をおこなう〝Déjà Vu〟というシステムを開発しています[1][1]。ユーザーは、計算機を用いて生成した何千枚もの画像のなかから、パスワードの代わりに〝pass portfolio〟と呼ばれる 5 枚の画像をあらかじめ選んでおきます。ユーザーが計算機にログインしようとすると、その 5 枚の画像を含む 25 枚の画像が提示されます。ユーザーが、そのなかから正しく 5 枚を選択できればログインに成功します。好きな画像のほうが暗号のような文字列よりも記憶に残りやすいので、こういった方法がうまく機能するということです。

私自身は Déjà Vu を使った経験がないのでよく分かりませんが、自分の好みに沿って選んだとはいっても、人工的な画像を憶えておくのは難しそうな気がします。仲のよい友人や、よく知っている場所の写真を並べたほうが効果的かもしれません。25 枚の中年男の写真のなかから、知っている〝オヤジ〟を 5 人選ぶ程度なら簡単にできそうです（あまり楽しい作業ではなさそうですが）。

図 1　Déjà Vu のログイン画面



能力にもとづく認証

脳のなかには、さして苦労せずに思い出せる情報もたくさん格納されています。たとえば、自転車に乗れる人は、倒れずに運転する処理をおこなうための情報が脳のどこかに収められており、ほとんど無意識のうちにそれを取り出せるはずです。残念ながら、自転車に乗れるかどうかは個人の識別にあまり役立ちませんが、特殊な能力をもつ人の場合はそれを認証に利用することも可能でしょう。たとえば、絶対音感をもっている人ならば、聴いた和音を即座に鍵盤で再現する能力を示すことで認証できます。あるいは、キーボードを素早く打てる人なら、提示された文字列をキーボードで入力する速度にもとづいて認証ができるかもしれません。

このような能力をもつ人は少数派なので、一般的な認証には使えません。しかし、ある人が特定のグループに属するかどうかのチェック程度であれば利用できそうです。たとえば、Oosaka や Nagoya などのローマ字に対応する漢字を書かせてみれば、日本人かどうかのいちおうの目安にはなるでしょう。

1 http://www.sims.berkeley.edu/~rachna/dejavu/

エピソード記憶にもとづく認証

能力にもとづく方法では細かい認証は困難なので、個人ごとに異なる人間の記憶を使うほうが望ましいのではないでしょうか。

意識的に思い出せる人間の長期記憶には、ものごとをシンボルとして憶えている〝意味記憶〟と自分の体験にもとづいて憶えている〝エピソード記憶〟があります。意味記憶とは、たとえば〝ペンギンは鳥である〟といった抽象的な知識であり、エピソード記憶とは、〝この前、秋葉原に行ったら編集長に会った〟というような体験にもとづく記憶です。

意味記憶は、試験に出してもおかしくないような一般的な知識であるのに対し、エピソード記憶は個人的な情報がほとんどです。シンボル的な意味記憶は憶えておくのが難しいうえに、複数の人間が共通にもつ知識である場合が多いのでどちらかといえば認証には不向きです。ところが、パスワードによる認証は、このような意味記憶にもとづく方式であり、これがもっとも大きな問題だと思われます。一方、エピソード記憶の大半は個人的なものであり、苦労せずに憶えることができ、かつ忘れにくいという特徴をもっています。シンボルを場所や身体に対応させて暗記する〝記憶術〟がありますが、これはエピソード記憶によって意味記憶を補填しようという試みです。

認証の場合も、人間のエピソード記憶をうまく使えば、パスワード方式のような問題は少なくなるはずです。たとえば〝この前、秋葉原で誰に会ったか?〟という質問ならすぐに答えられますし、しかも本人以外には答えることができません[2]。

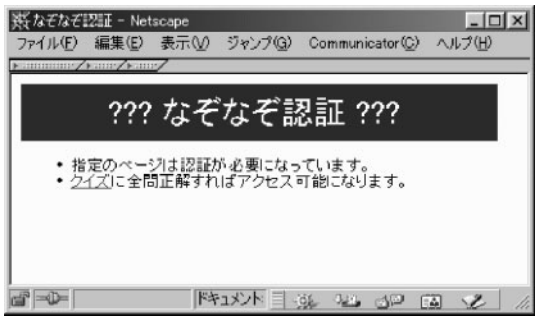
エピソード記憶にもとづく質問に対してすべて正しく答えられたかどうかによって認証すれば、より正確かつ簡単に認証をおこなえるようになる可能性があります。

長年のつきあいにもとづく認証

計算機と長年のつきあいがあれば、共有するいろいろな知識を用いた認証も可能になるかもしれません。10年かけて育てたロボット犬なら、飼い主を簡単に認証できるでしょう。クセや態度のようにシンボル化することが難しい情報でも、ある程度以上の期間のつきあいがあればうまく処理できるかもしれません。

2 個人的に強烈な意味記憶を使う方法も考えられます。たとえば、本人以外には秘密にしている〝初恋の人の名前〟などを使うことも考えられます。これは、もともとはエピソード記憶かもしれませんが……。

図2 認証が必要な場合の表示画面



なぞなぞ認証

このように、脳に蓄えられた能力や知識、経験などをうまく利用すれば、そのためだけによぶんな手間をかけずに安全な認証をおこなえそうです。

Webサービスのなかには、パスワードを忘れたときに備えて、本人しか解けないような問題と解答を登録できるようになっているものがあります。このようなサイトでは、たとえばパスワードを登録するときに〝母親の旧姓は?〟などという質問とその答を組にして登録しておきます。そして、万一パスワードを忘れてしまった場合でも、母親の旧姓を正しく答えられればパスワードを教えてもらうことができます。

このようなシステムは、パスワードは忘れる可能性があり、そのような場合に備えた対策を用意したという点では意義があります。しかし、それならばパスワードを登録したりするのではなく、最初から〝母親の旧姓は?〟に答えられたらサービスが受けられる仕組みにしておけばいいようにも思えます。

このような考えにもとづき、能力や知識、経験を問う〝なぞなぞ〟を解くことによって認証をおこなう「なぞなぞ認証」システムを作ってみました。

使用例

認証の対象とするURL(たとえば、http://www.csl.sony.co.jp/person/masui/diaryなど)をアクセスすると、そのページが直接表示されるのではなく、図2のような説明画面が表示されます。

図 3　なぞなぞの例

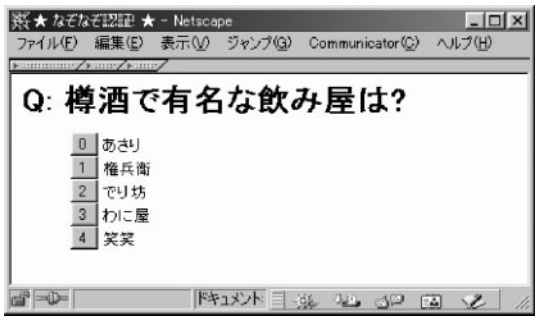


図 4　すべてに正しく解答した場合

リスト 1　.htaccess の記述例

```
CookieAuthMode  On
CookieName      diarycookie
CookieAuthSharedKey     SecretStringForMyDiary
CookieAuthDummyURI      http://www.my.domain/dummy.html
```

リスト 2　Cookie 生成スクリプト

```
#!/usr/bin/perl
use MD5;
$mydomain = ".my.domain";
$authname = "diarycookie";          # Cookie名
$secret = "SecretStringForMyDiary"; # 秘密文字列
$stp = time();

$digest = MD5->hexhash("$authname$stp$secret");
$cauth = "$authname:$stp:$digest";

print <<EOF;
Content-Type: text/html
Set-Cookie: $authname=$cauth; domain=$mydomain; path=/;

このようなCookieを取得するとアクセスが許可される
EOF
```

ここで ‵クイズ″をクリックすると、図 3 のように 5 択のなぞなぞが表示されます。

正しい答を示すボタンをクリックすると次の問題が表示されます。すべてに正しく答え終ると、図 4 のような画面が現れ、最初にアクセスしようとした URL が参照可能になります。

実装

なぞなぞ認証システムは、すべての問題に正解したときにサーバーが Cookie を発行し、その Cookie をもつブラウザからの要求に対してのみアクセスを許可する仕組みになっています。

Cookie によるアクセス制御のために、今回は沼田一成さんの作成した mod_cookieauth という Apache のモジュール[3]を使ってみました。

mod_cookieauth

mod_cookieauth モジュールを追加した Apache サーバーでは、.htaccess ファイルをリスト 1 のように記述しておくことで、特定の Cookie をもつブラウザに対してだけ閲覧を許可します。

たとえば、 ‵diarycookie″という名前をもつ Cookie は、リスト 2 のようなプログラムで生成します。

mod_cookieauth によるなぞなぞ認証の実装

mod_cookieauth を使ったなぞなぞ認証の設定ファイル quiz.pl とセットアップ・スクリプト setup.pl、CGI

3 http://www.guru.gr.jp/~issei/diy.html

プログラム quiz.cgi を末尾のリスト 3〜5 に示します。

quiz.pl には、なぞなぞ認証をおこなうディレクトリ、なぞなぞの問題と解答、秘密文字列などを記述しておきます。setup.pl を実行すると、quiz.pl の内容が参照されて .htaccess がセットされます。クイズの実行は quiz.cgi でおこないます。

なぞなぞ認証の長短

なぞなぞ認証には以下のような利点があります。

- パスワードを暗記する必要がない
  パスワードによる認証の安全性を高めるために複雑なパスワードを複数用意すると、憶えておくのが大変です。かといって、あらゆるシステムに同じパスワードを登録すると、盗まれたときの被害が大きくなってしまいます。これに対し、なぞなぞ認証ではそもそもパスワードが不要なので、認証の必要なサービスを気軽に使えます。
- 解くのが楽しく、解けなくてもあまり腹が立たない
  暗号のようなパスワード文字列を入力するより、なぞなぞを解くほうが人間的な感じがします。非公開の URL をアクセスしてしまったときなど、ユーザー ID やパスワードの入力を求めるダイアログがいきなり表示されるのは、あまり気分のよいものではありません。その代わりにユーモラスなクイズが表示されれば、それほど気分を害することもないのではないでしょうか。
- 問題をうまく作れば、特定のグループに公開できる
  関係者だけが知っている事柄を題材に問題を作れば、その人たちだけに情報を公開できます。たとえば、パーティーの写真を Web 上に置いておき、そのパーティーでの出来事に関する問題をなぞなぞにしておけば、出席者だけに写真を公開できるでしょう。

一方、今回の実装には次のような問題点もあります。

- 問題を解くのに時間がかかる
  脳内の情報を使っているかぎり、認証にある程度時間がかかるのは致し方ありません。ただし、インターフェイスを工夫すれば、パスワードと同程度の時間で認証をおこなうことも難しくはないでしょう。電車に乗るたびに改札口でなぞなぞを解かせるのは非現実的かもしれませんが、定期券を忘れた人のために〝なぞなぞ改札口〟を 1 つ用意しておいてもよいのではないでしょうか。
- よい問題を考えるのが面倒である
  本人にしか解けない問題を考えるのは、なかなか難しいものです。とはいえ、ちょっと変わった頭の体操と捉えれば、〝考えがい〟があるというものです。
- ブラウザ単位での認証であり、個人単位にはなっていない
  今回の実装はブラウザと Cookie を利用しているため、認証はブラウザ単位でおこなわれます。しかし、まったく異なる実装手法を使えば、個人単位の認証も可能だと思います。
- システムに問題も解答も知らせる必要がある
  今回の実装では、問題とその解答を記述したファイルをシステム上に置かなければなりません。したがって、本当にプライベートな情報に関する問題は使えないでしょう。ただし、ゼロ知識証明のような手法を使えば、システム上に解答を残さずに認証をおこなえるので、誰にも教えたくない秘密を問題として使うこともできます。

なぞなぞ認証は、最初に述べた各種の認証方式と組み合わせることもできます。自分の行動をもとになぞなぞを自動生成することも可能でしょうし、Déjà Vu のように画像や写真を問題に使うこともできるでしょう。

おわりに

〝なぞなぞで認証する〟などと聞くと、バカバカしく思えるかもしれません。しかし、素っ気ないパスワード認証方式とパスワード管理の煩雑さにくらべれば、まだマシではないでしょうか。ネットワークの利用者は今後ますます増えていくでしょう。その場合、なぞなぞにかぎらず、楽しく効率的な認証方式が重要になってくると思います。

[参考文献]

[1] Rachna Dhamija and Adrian Perrig, “Déjà Vu: A user study using images for authentication”, In *9th Usenix Security Symposium*, August 2000

リスト 3　設定ファイル (quiz.pl)

```
# なぞなぞ認証設定ファイル
$domain = ".csl.sony.co.jp";
$baseURI = "http://www.csl.sony.co.jp/person/masui";
$basedir = "/user/masui/person/masui";
$expiredays = 30;

$curURI = "$baseURI/quiz/diary/index.html";
$secretdir = "$basedir/diary";
$secretURI = "$baseURI/diary";
$authname = "diaryauth";
$secretkey = "SecretKeyForMasuisDiaryNewXXXX";

@questions = (
        [ '樽酒で有名な飲み屋は?', 0,
          ['あさり', '権兵衛', 'でり坊',
           'わに屋', '笑笑']
        ],
        ['増井の特技は?', 1,
          ['自転車に乗りながら饂飩を食べる',
           '口笛を吹きながら鼻歌をならす',
           'タイピングしながらメモをとる',
           'ピアノをひきながら読書する',
           '寝ながらプログラムを書く']
        ],
        ......
);
1;
```

リスト 4　セットアップ・スクリプト (setup.pl)

```
#!/usr/bin/perl
require 'quiz.pl';

open(out,"> $secretdir/.htaccess") || die "Can't open $secretdir/.htaccess";
print out <<EOF;
CookieAuthMode On
CookieAuthName $authname
CookieName $authname
CookieAuthSharedKey $secretkey
CookieAuthDummyURI $curURI
EOF
close(out);
```

リスト 5　なぞなぞ CGI (quiz.cgi)

```
#!/usr/bin/perl

use MD5;
require 'cgiarg.pl';
require 'quiz.pl';

$time = time;

# CGI呼出しの引数を変数に代入
# $nq: これまでに出した問題の数 (例:2)
# $qs: これまでの問題番号リスト (例:9,3,2)
```

```
# $as: 前回までの解答リスト      (例：2,0)
# $pa: 前回の解答                (例：3)

&getcgiarg;

if($nq > 0){
  @qs = split(/,/,$qs);
  @as = split(/,/,$as);
  push(@as,$pa);
}

# $nask 解答後なら判定
$nask = 5;
&quiz if $nq < $nask;
for($i=0;$i<$nask;$i++){
  &fail if $as[$i] != $questions[$qs[$i]]->[1];
}
&success;

sub quiz { # 出題
  # まだ出題していない問題を選ぶ
  for (@qs){ $qs{$_} = 1; }
  srand(time);
  for(;;){
    $qno = int(rand($#questions+1));
    last if ! $qs{$qno};
  }
  push(@qs,$qno);
  $qs = join(',',@qs);
  $as = join(',',@as);
  $q = $questions[$qno]->[0];
  $a = $questions[$qno]->[2];
  $nq++;

  print <<EOF;
Content-type: text/html

<html>
<head><title>★ なぞなぞ認証! ★</title></head>
<body bgcolor=white>
<h1>Q: $q</h1>
<form method="post" action="quiz.cgi?time=$time">
<blockquote>
<input type=submit name=pa value=" 0 "> $a->[0]<br>
<input type=submit name=pa value=" 1 "> $a->[1]<br>
<input type=submit name=pa value=" 2 "> $a->[2]<br>
<input type=submit name=pa value=" 3 "> $a->[3]<br>
<input type=submit name=pa value=" 4 "> $a->[4]<br>
</blockquote>
<input type="hidden" name="nq" value="$nq">
<input type="hidden" name="qs" value="$qs">
<input type="hidden" name="as" value="$as">
</form>
</body>
</html>
EOF
```

```
  exit;
}

sub success { # 成功通知 / Cookie設定
  $expiresec = time + $expiredays * 24 * 60 * 60;
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
    gmtime($expiresec);
  $expirestr = sprintf("%s, %02d-%s-%02d %02d:%02d:%02d GMT",
    ('Sunday','Monday','Tuesday','Wednesday',
     'Thursday','Friday','Saturday')[$wday],
    $mday,
    ('Jan','Feb','Mar','Apr','May','Jun',
     'Jul','Aug','Sep','Oct','Nov','Dec')[$mon],
    $year % 100,
    $hour,$min,$sec);

  $digest = MD5->hexhash("$authname$expiresec$secretkey");
  $cauth = "$authname:$expiresec:$digest";

  print <<EOF;
Content-type: text/html
Set-Cookie: $authname=$cauth; domain=$domain; path=/; expires=$expirestr;

<html>
<head><title>認証成功</title></head>
<body bgcolor=white>
認証に成功しました! ブラウザに
$expiredays
日間のアクセス権が設定されました。
<p>
<a href="$secretURI">目的ページにジャンプ</a>
</body>
</html>
EOF
  exit;
}

sub fail { # 失敗通知
  print <<EOF;
Content-type: text/html

<html>
<head><title>認証失敗!</title></head>
<body bgcolor=white>
認証に失敗しました!!^&@#*^&)%?
<p>
<a href="./quiz.cgi?time=$time">もう一度挑戦</a>
</body>
</html>
EOF
  exit;
}
```