**SCIS 2008** The 2008 Symposium on Cryptography and Information Security
Miyazaki, Japan, Jan. 22-25, 2008
The Institute of Electronics, Information and Communication Engineers

# フィッシング防止のための HTTP パスワード相互認証プロトコル

# PAKE-based mutual HTTP authentication for preventing phishing attacks

鈴木　博文* Hirofumi Suzuki　大岩　寛† Yutaka Oiwa　高木　浩光† Hiromitsu Takagi　渡辺　創† Hajime Watanabe

**あらまし**　偽サイトでパスワードや個人情報を詐取するフィッシングの問題を抜本的に解決する新しいプロトコル「HTTP Mutual 認証」を提案する。これは、サーバとクライアント間でパスワードを用いた真の相互認証を実現するプロトコルであり、ユーザインターフェイスの改良と併せてフィッシングを防止する。設計においては現実の Web アプリケーションの利用形態を考慮し、実用プロトコルとして現状の認証手段の置き換えが容易に可能となるよう配慮した。

**キーワード** フィッシング、パスワード、相互認証、PAKE、HTTP

## 1　はじめに

インターネットではここ数年、フィッシングと呼ばれる手口で、個人情報やパスワード等を詐取しようとする悪質な行為が横行し、社会問題となっている。一部のブラウザは偽サイト検知機能を搭載するなどして対応しているが、抜本的な解決策ではない。

我々は HTTP 認証を拡張した Mutual 認証を Web システムに導入することで、この問題を抜本的に解決する方法を提案する。本提案手法の設計にあたっては Web アプリケーションの利用形態を考慮し、現状の認証手段の置き換えとしての利用を実現できるよう工夫した。

本論文では、最初にフィッシングの手口と既存の対策を整理し、提案する解決法の特徴を述べる。続けて、設計したプロトコルの概要と設計の趣旨を説明し、本認証の利用に際して必須となるユーザインターフェイス設計について述べる。

* Yahoo 株式会社 オークション事業部, 〒106-6182 東京都港区六本木 6-10-1, 六本木ヒルズ森タワー, Yahoo Japan Corp, Auction division, Roppongi Hills Mori Tower, 6-10-1 Roppongi, Minato-ku, Tokyo 106-6182, JAPAN (hirsuzuk@yahoo-corp.jp).
† 独立行政法人産業技術総合研究所情報セキュリティ研究センター、〒101-0021 東京都千代田区外神田 1-18-13 秋葉原ダイビル 1102, Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, 1102 Akihabara Daibiru, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021 JAPAN.

## 2　フィッシングの類型

フィッシングの手口は、次のように分類することができる。

(0) 認証を要求せずに、直接クレジットカード番号や個人情報等を入力させ、これを騙し取る。
(1). 認証を要求し、入力されたユーザ ID とパスワードを騙し取る
(2). 認証を要求するが、入力したパスワードによらず認証が成功したと見せかけ、続く画面でクレジットカード番号や個人情報等を入力させ、これを騙し取る[7]。
(3). 認証を要求し、入力されたパスワードを正規サイトに中継し、パスワードが正しいときだけ、(0)、(1)と同様にパスワードやクレジットカード番号等を騙し取る。
(4). 認証を含むすべての通信内容を正規サイトに中継し、通信内容の窃取、セッション識別情報を窃用したセッションハイジャック、リクエスト内容の改竄等を行う。

初期のフィッシングは、(0)、(1)の単純な手法が主なものであった。ユーザにフィッシングという手口が認識されるにつれ、(0)のような単純な偽サイトではユーザを騙せなくなったため、(2)の手口が流行するようになった。また、(1)や(2)の手口に対して、一回目に嘘のパスワード

を入力して確認する自衛策があることから、最近では(3)のような手法が現れている。

また、金融機関などで二要素認証の導入が進んだことにより、パスワードを詐取しただけでは目的が達し得なくなったことから、(4)の手口が現実に現れている [8]。

こうしたフィッシングの被害に遭わないための基本的な対策は、常に接続したサイトのドメイン名を目視で確認することである。しかし、実際には一般のユーザがドメイン名を記憶し忘れずに確認することは現実的ではなく、偽サイトに紛らわしいドメイン名が使われている場合などには、経験豊富なユーザでも見逃してしまう可能性がある。

# 3 既存の対策とその問題点

こうした問題に対し、これまでにも様々な技術的対策が提案されてきた。それらは次のように分類できる。

## 3.1 ホワイトリスト方式

ホワイトリスト方式とは、なんらかの方法で信用してよいサイトリストをあらかじめ提供し、アクセス時に確認するものである。

たとえば Extended Validation SSL 証明書 (EV SSL 証明書) [4]はその一つであり、CA/Browser フォーラムが定めた厳格な審査基準を認証局が遵守し、その基準を満たすサイトに対してのみ発行されるサイト証明書である。すなわち、これは認証局が選別したサイトが偽サイトではないことを保証する方式である。

EV SSL 証明書を導入しているサイトにアクセスすると、ブラウザが通常の SSL 証明書と区別して特別な表示をする。たとえば、Microsoft Internet Explorer 7 で、EV SSL 証明書を採用しているサイトにアクセスするとアドレスバーが緑に変わる。ユーザは利用時に、このブラウザの特別な表示を確認してから個人情報等を入力することで、フィッシングの被害を防ぐことができる。

一般にホワイトリスト方式の問題点として、審査基準が厳しければ正規のサイトがリストに加わることができないケース、逆に基準が緩ければ偽サイトが排除できなくなるケースが起きてしまうことが挙げられる。実際 EV SSL 証明書は、優良なサイトであっても CA/Browser フォーラムの審査基準を満たすことができずに、発行されないことがある。

## 3.2 ブラックリスト方式

ブラックリスト方式とは、なんらかの方法で偽サイトのリストを提供し、アクセス時に確認するものである。たとえば、Microsoft Internet Explorer 7 や Mozilla Firefox 2 は、偽サイトとして報告されている URL へのアクセスをブロックする機能を搭載している。

しかし、完全なブラックリストを作ることは原理的に不可能であり、現実にも新たに登場する膨大な偽サイトに対応してリストの更新が追いついていない。たとえば、Mozilla および Microsoft が 2006 年に行った調査 [9] [10]によれば、既知の偽サイトにおいて上記の二ブラウザが偽サイトとして検知できた割合は 66%から 81%にとどまっている。

## 3.3 パスワードマネージャ

パスワードマネージャは、ユーザ名やパスワードをブラウザに保存しておき、特定のサイトでそれらの情報が必要になった場合、自動的に入力する機能である。一度ログインしたサイトでなければ、パスワードが自動入力されることがないので、パスワード入力を完全にパスワードマネージャに任せておけば、偽サイトの検出機能として用いることができる。

ただし、パスワードを登録したブラウザのみでしか利用することができない機能であるため、外出先など複数のコンピュータを用いる場合には有効でない。

## 3.4 アドレス確認の補助ツール

例えば、Firefox のアドオンである petname tool [6]は、ユーザが任意のサイトに対して、それぞれに名前を関連付けることが出来る機能である。

ユーザが一度正規サイトであることを確認したサイトに名前を付けておくと、以降そのサイトにアクセスする毎に、名前が緑色で囲われてツールバーに表示される。名前が登録されていない場合は、untrusted という文字が表示され緑色にならないので、紛らわしいドメインを使った偽サイトを訪れたとしても、本物と区別することが可能になる。これにより、ユーザは緑色になっていることを確認してからユーザ名、パスワード等の個人情報を入力することを徹底すれば、フィッシングの被害を防ぐことができる。

しかしこのようなツールも、パスワードマネージャ同様に、名前を登録したブラウザ以外での利用が出来ない。

## 3.5 PwdHash

PwdHash [5] は、入力されたパスワードを接続先サイトのホスト名（FQDN）とともにハッシュ変換することで、そのサイト固有のパスワードを生成し自動で入力するものである。万が一ユーザが偽サイトにパスワードを送信したとしても、正規ホスト向けのハッシュ値と異なる値が送信されるので、前述の(1)、(3)、(4)の攻撃を防ぐことができる。

しかし、送信されたパスワードにかかわらず認証の成功を偽装する(2)の攻撃を防ぐことはできない。

また、単純にハッシュ関数を用いるこの方法は、オフライン辞書攻撃に対して脆弱である。今日の計算機では、平均的な８文字程度のパスワードを用いた場合、オフライン辞書攻撃によって現実的な時間でハッシュ値からパスワードを推測することが可能である。このような攻撃を防ぐためには、ユーザは非常に長いパスワードをサーバに設定しておく必要がある。しかし NIST [12, Appendix A] によれば、ユーザが自ら設定したパスフレーズのエントロピーは 40 文字でも高々62 ビット相当程度と評価されており、記憶可能な長いパスワードでオフライン攻撃を防ぐことは現実的なアプローチとはいえない。

### 3.6 TLS クライアント認証

パスワード認証の代わりに、公開鍵暗号に基づく TLS クライアント認証を行うことで、上述の(1)、(3)、(4)の攻撃を防ぐことができる。TLS の仕様では、サーバ認証とクライアント認証は独立して行われ、サーバ認証は依然として通常のサーバ証明書を用いておこなわれるため、(2) 同様にクライアント認証の成功を偽装する攻撃が可能である。

また、パスワードマネージャ同様に、秘密鍵をインストールしたコンピュータ以外で利用できない。

### 3.7 問題点のまとめ

以上の手法を大別すると、3.1 と 3.2 はインターネット全体の正規ホストと偽サイトの集合を管理しようとするアプローチでありリストの不完全さが問題となる。3.3 と 3.4 はクライアント側にある過去の記録を元に正規ホストを識別するアプローチであり、同じコンピュータでしか利用できないことが欠点となる。3.5 と 3.6 は暗号的手段でクライアント認証を強化するアプローチであり、サーバがクライアントの正当性を確認しない(2)の攻撃には対処できない。

## 4 新しい解決法の提案

以上の問題整理に基づき、我々は以下の特性を持つ新しい「HTTP パスワード相互認証プロトコル」を設計した。

(a) 簡単に利用できる
　ユーザが慣れ親しんだパスワードのみを使用し、鍵管理などの必要がない。

(b) どこでも利用できる
　ローカルストレージを使わず、自分でリストを管理する必要がない。

(c) 一般的
　第三者によって管理されるリストに依存しない。

(d) 相互認証
　クライアントとサーバが相互に認証することで、偽サーバが認証成功したふりをできないようにする。

(e) オフライン攻撃にも耐性を持つ
　偽サイトと通信した場合でも通信データからパスワードを復元できないことが情報論的に保証されている。

(f) Web システムとの親和性
　既存の認証方式や Web アプリケーション設計との互換性に配慮しており導入が容易である。

次章で我々の提案するプロトコルは、ユーザが入力したパスワードのみを用いて、前述(1)から(4)の攻撃を防ぐことができる。

なお、前述(0)の攻撃は、ある種ソーシャルアタックに分類される問題であり、インターネットのようにオープンなネットワークにおいては技術的に完全な解決は困難である。短期的には、ブラックリスト方式などのアドホックな手法に頼らざるを得ないと考えられるが、提案プロトコルが十分に普及すれば、認証済みサイト以外には個人情報を入力しないという簡単なリテラシーによって解決しうる。

## 5 HTTP パスワード相互認証プロトコル

### 5.1 PAKE を Web 認証向けに利用

本プロトコルは、ISO/IEC 11770-4 [1] で定義されている PAKE (Password Authenticated Key Exchange) の一種である Key Agreement Mechanism 3 (KAM3) をベースプロトコルとし、Web 向けに改良したものである。本プロトコルでは、クライアントとサーバが同じパスワードを保持していることを相互に確認することで認証が成功する。つまり、認証が成功すれば、接続先相手は以前にユーザがパスワードを通知したサイトであり、ユーザの本当に意図したサイトであることを意味する。

認証の過程においてパスワードを直接送信することがなく、パスワード情報が盗聴者および中継者によって盗まれることが無い。また、窃取した情報を元にオフライン攻撃による全探索を行った場合でも、正規のパスワードに関する情報を入手することが不可能になっている。

なお、本論文ではプロトコルの概要についてのみ述べる。具体的なプロトコルについては、Internet Draft [3] を参照されたい。

### 5.2 ホスト名を使ったフィッシング検知

通常の PAKE の利用では、認証と同時に両者で共有される秘密情報から共通の鍵を生成し、それを用いて暗号化通信を行うことで中間者攻撃を防止するが、本プロトコルでは、主に HTTP プロトコルおよび既存の Web システムとの親和性の理由から、暗号化には既存の TLS の枠組みを用いる一方で、PAKE を HTTP プロトコルのレイヤにおいて認証のみに用いることとした。これにより起こる偽サイトから正規サイトに認証トークンを転送する攻撃の可能性を防ぐため、パスワードとホスト名を結びつけ、接続先を識別するように PAKE を変形した。この変形により、たとえ中継攻撃を行う偽サイトに接続してパスワードを入力したとしても、正規サイトとはホスト名が異なるため認証が失敗することになる。また用いる PAKE の性質から、パスワード自体のデータが偽サイトに漏れる事もない。

### 5.3 具体例

初回の通信では、クライアントとサーバ間で相互認証が成立するまでに、3 往復のメッセージ交換が発生する。

(1) 認証されていないユーザ（のクライアント）からのリクエストをサーバが受けた場合、HTTP401 ステータスをクライアントに返す。

```
GET / HTTP/1.1
Host: www.example.com

HTTP/1.1 401 Authentication required
WWW-Authenticate:  Mutual
   algorithm=iso11770-4-ec-p256,
   validation=host,
   realm="Protected Contents", stale=0
Content-Type: text/html; charset="ISO-8859-1"
Content-Length: xxx

<html>……
```

(2) クライアントはユーザに、ユーザ名とパスワードの入力を要求する。ユーザがユーザ名、パスワードを入力後、クライアント内部では、ホスト名、realm 値、ユーザ名、パスワードを組み合わせ Hash した値である π を生成する。この π を PAKE における weak secret として ISO/IEO1770-4 の仕様にしたがって $w_A$ を生成し、2 回目のリクエストをサーバに送信する。

サーバは、自身に保存されているパスワードを検証するための値 v をつかって $w_B$ を計算し、新たに生成したセッション ID sid と共に再度 HTTP401 を返す。

```
GET / HTTP/1.1
Host: www.example.com
Authorization: Mutual
   algorithm=iso11770-4-ec-p256,
   validation=host, user="foobar",
   wa=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

HTTP/1.1 401 Authentication required
WWW-Authenticate: Mutual sid=yyyyyyyy,
   wb=zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz,
   nc-max=1024, nc-window=64, time=300,
   path="/"
Content-Length: 0
```

(3) クライアントは、受信した $w_B$ の値とパスワード π から認証値 $o_A$ を生成し、3 回目のリクエストを送信する。サーバはその $o_A$ を検証し、正しい値であればユーザが正しいパスワードを知っていることを意味するので、もう 1 つの認証値 $o_B$ を生成し、最後のレスポンスに $o_B$ を付加し、HTTP200 ステータスとともにクライアントに返す。

クライアントは、必ずその $o_B$ を検証し、正しい値であれば接続先のサーバは接続を意図したサーバであることを確認する。

```
GET / HTTP/1.1
Host: www.example.com
Authorization: Mutual
   algorithm=iso11770-4-ec-p256,
   validation=host, user="foobar",
   sid=yyyyyyyy, nc=0, oa=wwwwwwwwwwwwwwww

HTTP/1.1 200 OK
Authentication-Info: Mutual sid=yyyyyyyy,
   ob=vvvvvvvvvvvvvvvv
Content-Type: text/html; charset="ISO-8859-1"
Content-Length: xxx

<html>……
```

### 5.4 既存の Web システムとの親和性のための改良

本プロトコルでは、基本的な認証の機能に加え、既存の Web アプリケーション・システムとの親和性を高めるための HTTP 認証機構 [2] の改良を加えている。

現状の Web アプリケーション認証は、Basic 認証、Digest 認証をほとんど利用しておらず、代わりに各サイトが独自に用意した Form 認証が主流となっている。Basic 認証、Digest 認証が普及していない原因としてはいくつかの理由が考えられるが、主として次の要因が大きな障害が挙げられる。

1. 認証の状態をサーバ側から管理できない。もっとも大きな問題点としては、サーバ側からの要求によりログイン状態を解除する機能（ログアウト機能）を実装し得ない。
2. 通常のサーバでは特定の URL に対して認証の有無を固定的に割り当てる必要があり、最近の Web サービスに多いゲスト向け情報と認証ユーザ向け情報を同じ URL で提供する構成をとることが困難である。

Form 認証はフィッシングに極めて脆弱なため、フィ

ッシング問題の解決のためにはこれらを Mutual 認証に置き換えることが望ましいが、全く新しい認証方式を提案したとしても、一般的に受け入れられることがなければ意味がない。そのため、我々は Mutual 認証方式と同時に、これらの問題点を解決するための HTTP 認証の拡張をあわせて提案する。

具体的には前者の問題に対しては、サーバ側からブラウザに対し、ログアウトの要求を明示的に示すヘッダの拡張を導入した[1]。後者の問題に対しては、サーバ側がクライアント側に「任意の認証」の可能性を提示するため、HTTP の成功レスポンスコード 200 の場合において追加的に用いる Optional-WWW-Authenticate ヘッダを用意した。

```
HTTP/1.1 200 OK
Optional-WWW-Authenticate: Mutual
algorithm=iso11770-4-ec-p256,
validation=host,
  realm="Protected Contents", stale=0
Content-Type: text/html; charset="ISO-8859-1"
Content-Length: xxx

<html>……
```

Mutual 認証をサポートしているクライアント（ブラウザ）が、この Optional-WWW-Authenticate を受け取った場合、通常のコンテンツを表示しつつ、認証情報入力欄をアクティブな状態にし、ユーザの任意で認証を開始できるようにする。

### 5.5 今後の検討課題

本プロトコルの基本的な認証部分は完成しているが、主に既存 Web システムとの親和性についてはいくつかの検討の余地がある。例えば、銀行サイト等で行われているログイン中のパスワードの再確認、同一ドメイン内の複数ホストにおけるログイン状態の共用（シングル・サイン・オン）、ワンタイム・パスワード・トークンの利用等は現状のプロトコルではサポートされていない。これらの要求の可能性については、Yahoo! JAPAN サイトでの実証実験等を通じて必要性を検討していく。

## 6 ユーザインターフェイス

5.1 と 5.2 で述べたように、本プロトコルは、ユーザが誤って偽サイトでパスワードを入力しても攻撃者にパスワードの情報が漏れないことを実現するものである。しかし、パスワードを入力する欄が攻撃者の仕掛けた偽のものであっては元も子もない。従来の HTTP 認証では、認証情報の入力欄は多くのブラウザにおいてモーダルなダイアログボックスに設けられているが、ダイアログボックスは通常コンテンツ領域の上に重なって現れるものであるため、偽サイトがダイアログボックスを模したコンテンツを表示した場合、ユーザは騙されてそこに認証情報を入力してしまう危険がある。

この問題を解決するため、今回 Mutual 認証のブラウザ実装をするにあたっては、認証情報入力欄をブラウザのアドレスバーの隣に配置することとした。Mutual 認証を要求しないページでは入力欄は表示されず、Mutual 認証でログインが可能なサイトにアクセスすると図 1 のように入力欄が現れる。
ログインが成功すると、図 2 のように緑の表示となりログイン中のユーザ名が表示される。これは、Mutual 認証が正常に完了していることを示すものであり、この Web サイトが偽サイトでないことを示す重要な機能である。

このようなインターフェイスで構成された Mutual 認証機構がブラウザに標準搭載するようになり、Web アプリケーションのログイン機能が Mutual 認証で実現されるようになれば、ユーザは次の使用法を徹底することで、フィッシングに騙されることを防止できる。

- ユーザ名、パスワードは必ずアドレスバー隣の入力欄に入力する。
- クレジットカード番号や個人情報を入力するときは、Mutual 認証でログイン中であることを必ず確認してから行う。

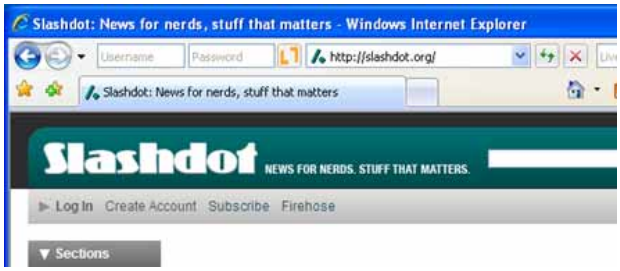

図 1: ログインが可能な状態の画面

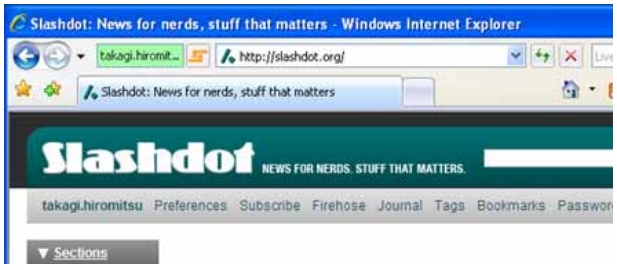

図 2: ログイン成功時の画面

[1] もちろんブラウザ側がログアウト要求を無視する実装を行う可能性は存在するが、これは本拡張に限った問題ではなく、既存の Form＋Cookie の認証でも自動的にパスワードを送出するパスワードリマインダーなどで同様の「機能」を実現できる。

## 7 プロトコル設計に関する検討

本プロトコルのような認証を設計するに当たっては、HTTP レイヤ（暗号化通信を行う場合には、TLS との組み合わせ）で実現する方法の他に、TLS などの暗号化通信レイヤを拡張して実現する方法も考えられる。TLS-SRP [12]は、TLS に SRP という PAKE 鍵交換の一種を導入し、パスワード認証つき暗号化通信路を実現している。元来 TLS-SRP は IMAP などの認証に用いることを想定しており、フィッシング防止を目的としたものではないが、我々の提案するプロトコルで実現した機能のいくつかは TLS-SRP によっても実現することが可能である。しかし検討の結果、TLS-SRP の Web での利用にはいくつかの難点があることが判明している。

まず、ユーザ認証が TLS のハンドシェークの初期の段階で行われるため、1 ホスト内にゲスト用領域を含む複数の認証領域を設定するなど既存の Web アプリケーションが要求するサイト設計を満たすことができないため、大規模サイトへ導入は困難であると考えられる。

また、HTTP は 1 つの接続上で複数のリクエストが送受信されるにもかかわらず、TLS-SRP が接続単位で認証を行うため、Web アプリケーション向けの拡張を実現するためには結局 HTTP レイヤに暗号化の制御を行う機構を追加する必要が生じ、TLS レイヤでの認証とページ単位アクセス制御との関係を維持するために不必要に複雑な制御が必要になると考えられる。

以上のような理由から、本提案では暗号化レイヤでの認証を拡張する設計は避け、現在のような HTTP レイヤでの認証を採用した。また、HTTP レイヤでの認証設計は、暗号化アクセラレータなどの既存の通信機器と組み合わせる場合にも利点が多い。我々の提案プロトコルでは、これらの機器と組み合わせた際の動作についても考慮している。

## 8 今後の展開

### 8.1 実証実験

2008 年 4 月頃を目処に、Yahoo Japan! オークションサイト上で公開の実証実験を行う予定である。その実験には、一般からの参加者を募り HTTP パスワード相互認証プロトコルに対応したブラウザもしくはブラウザプラグインを配布する。

### 8.2 オープンソースコミュニティへの働きかけ

本プロトコルの普及にむけて、現在開発中のブラウザプラグイン（Firefox アドオン)と、サーバモジュール（Apache)のソースコードを、それぞれのオープンソースコミュニティに提供する予定である。

### 8.3 標準化

本プロトコルの標準化（RFC）を目指すため、IETF (Internet Engineering Task Force) に 2007 年 11 月 Internet Draft [3] を提出した。

## 参考文献


[1] International Organization for Standardization, "ISO/IEC 11770-4:2006: Information technology – Security techniques – Key management – Part 4: Mechanisms based on weak secrets", 2006.

[2] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[3] Oiwa, Y., Watanabe, H., Takagi, H., Suzuki, H., "Mutual Authentication Protocol for HTTP", https://datatracker.ietf.org/drafts/draft-oiwa-http-mutualauth/, November 2007.

[4] CA/Browser Forum, "About EV SSL Certificates", http://www.cabforum.org/certificates.html.

[5] Ross,B., Jackson, C., Miyake, N., Boneh, D., C. Mitchell, J., “Stronger Password Authentication Using Browser Extensions.”, Proceedings of the 14th Usenix Security Symposium, 2005.

[6] Close, T. “Petname Tool: Enabling web site recognition using the existing SSL infrastructure”, W3C Workshop on Transparency and Usability of Web Authentication, 2006

[7] マイコミジャーナル, “UFJ銀行を騙るフィッシングメールが出回る”, 毎日コミュニケーションズ, 2005年3月15日, http://journal.mycom.co.jp/news/2005/03/15/006.html

[8] Brian Krebs, “Citibank Phish Spoofs 2-Factor Authentication”, 2006年7月10日, http://blog.washingtonpost.com/securityfix/2006/07/citibank_phish_spoofs_2factor_1.html

[9] Mozilla Foundation, "Firefox 2 Phishing Protection Effectiveness Testing," November 14, 2006, http://www.mozilla.org/security/phishing-test.html.

[10] Tony Chor, "Anti-Phishing Accuracy Study," IEBlog, Microsoft Cooperation, September 28, 2006. http://blogs.msdn.com/ie/archive/2006/09/28/774513.aspx.

[11] National Institute of Standards and Technology, "Electronic Authentication Guideline," NIST Special Publication 800-63, April 2006. http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[12] Taylor, D., Wu, T., Mavrogiannopoulos, N., Perrin, T., "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, 2007.