Web サーバにおける証明書を用いた安全なユーザ認証の実現

服部 裕之 [*]

1999.7.19

Abstract

Web ページのアクセス制御の一つに、ブラウザ側から提示されたユーザの証明書を用いて認証を行う方法があるが、従来の Web サーバは、証明書の発行元 (CA - Certification Authority) の確認を、リアルタイムに行う機能が不完全であった。

これを解決する為に、Web サーバと CA 間においてリアルタイムに証明書の検証を可能にするシステムを開発した。

キーワード ： Web サーバ、アクセス制御、ユーザ認証、公開鍵暗号方式、証明書、CA、ICAP

1 はじめに

Web ページのアクセス制御を実現するには、主に以下の 3 つの手法がある。

1. パスワードを用いたアクセス制御
2. クライアントコンピュータの IP アドレスを用いたアクセス制御
3. 証明書を用いたアクセス制御

1. の方法では、あらかじめ Web ページへのアクセスを許可するユーザ名、パスワードを Web サーバに登録する。Web サーバは、ユーザがアクセス制限のあるページへのアクセスを試みた時に、ユーザ名およびパスワードの入力を求める。入力されたパスワードが正しいものであるならば、アクセス制限のある Web ページをユーザに提示する。

2. の方法では、あらかじめ Web サーバに、Web ページへのアクセスを許可するコンピュータの IP アドレスを登録する。Web サーバは、ユーザがアクセス制限のあるページへのアクセスを試みた時に、ユーザが使用しているコンピュータが登録されている IP アドレスを持っているのならば Web ページをユーザに提示する。

3. の方法では、あらかじめ Web サーバに、Web ページへのアクセスを許可するユーザが持っている証明書の情報を登録する。登録する情報は、証明書情報の一部である所有者名 (Subject フィールド) などである。Web サーバは、ユーザがアクセス制限のあるページへのアクセスを試みた時に、ユーザに対して証明書の提示を求める。そして、その証明書の内容が偽造されたものでなく、かつ、証明書の所有者名が Web サーバに登録されているものであるならば、 Web ページをユーザに提示する。

1. は Web ページへのアクセス制御手法として一般的なものであるが、

- パスワードがネットワーク上を流れてしまうのでセキュリティ上の不安がある。

---

[*] 明治大学情報科学センター / 情報システム管理課 (E-mail : hhat@isc.meiji.ac.jp)

- パスワードを管理するコストがかかる。

などの問題がある。

また 2. は、Web ページへのアクセスを、社内や学内ネットワークからのアクセスのみに許可したい場合などに一般的な手法である。しかし、

- IP アドレスを元にしている為、あくまでクライアント単位での制限しかできない。
- クライアントの IP アドレスが偽造された場合に無力である。

という問題がある。

これらに対し、3. は証明書と秘密鍵のデータをフロッピーディスクや IC カードに格納し携帯することによって、1. や 2. の問題を回避することができる。この為、証明書を用いたアクセス制御技術は、電子商取引などの分野で注目を浴びており、各所で実証実験が行われている [6] [7] [10]。

2 証明書を用いた Web サーバのアクセス制御技術

2.1 証明書とユーザ認証

証明書を用いたユーザ認証の基本的理論となっているのは、公開鍵暗号方式である。公開鍵暗号方式は、メッセージの暗号化に使う鍵と復号化に使う鍵を互いに異なるものにできる。この性質を利用し、一方の鍵を公開（公開鍵）し、もう一方の鍵を秘密（秘密鍵）にすることができる。

公開鍵暗号方式を用いた暗号通信を行なう際、相手の公開鍵は、ネットワークを利用して入手するのが一般的である。ところが、ネットワーク上の通信には、なりすまし、改竄など、セキュリティ的な問題がある為に、通信相手の公開鍵が正真正銘本人のものであるかどうかを、受信側で証明できるか否かは、重要な問題である。

この問題に対する解決策の一つは、第三者機関によって、ある者の公開鍵が偽造されたもので無いことを証明するという仕組みを用意することである。これは、ある者の公開鍵を、第三者機関のもつ 秘密鍵で暗号化し、それをその者の「証明書」として発行することによって実現できる。

この第三者機関のことをインターネットでは証明書発行局 CA (Certification Authority) と称している。

証明書を用いたユーザ認証を行う場合には、ユーザ（認証される側）に秘密鍵と公開鍵を持たせ、Web サーバ（認証する側）はユーザがその秘密鍵を本当に持っていることを確認することによって行う。

具体的な手順は次の通りである（図 1 参照）。

1. ユーザ側（認証される側）でテキスト（例えば乱数など）を生成する。
2. そのテキストをユーザ側の秘密鍵を用いてディジタル署名を行い、サーバへ送付する。
3. ユーザの証明書（公開鍵）をサーバへ送付する。
4. サーバ側は証明書の公開鍵情報を用いて、送付されたテキストの署名検証を行う。
5. 署名検証が正常に行えた場合、サーバの通信相手であるユーザは、証明書の正しい所有者であることが確認できる。

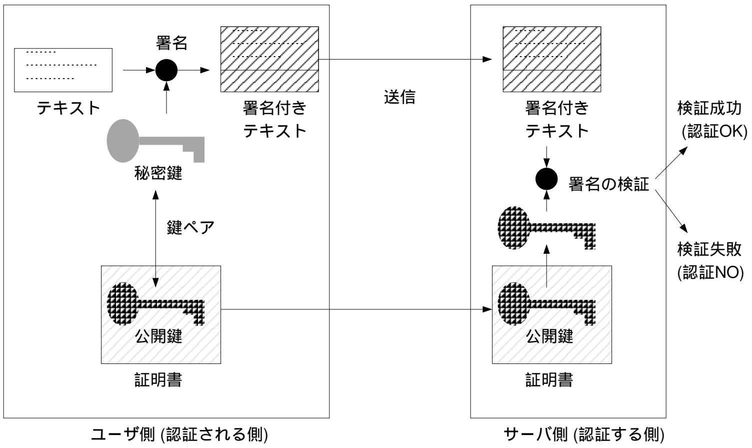
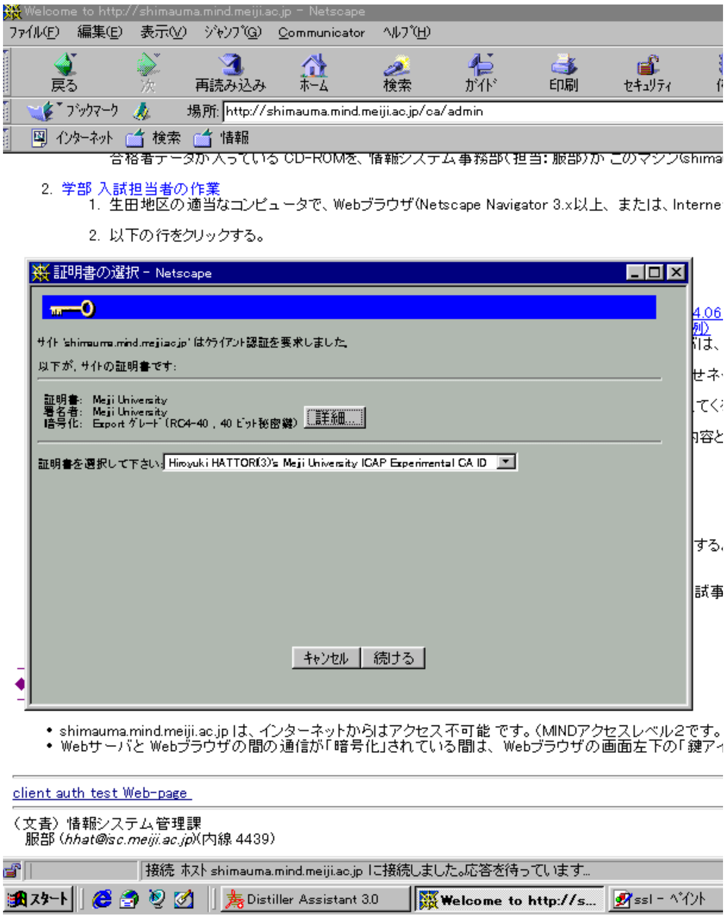


図 1 証明書を用いたユーザ認証の仕組み

この原理を応用したのが、安全性を考慮した Web サーバへのアクセスに広く用いられている SSL(Secure Socket Layer) による認証技術である。図 2 に SSL を用いたユーザ認証の様子を示す。



場所: http://shimauma.mind.meiji.ac.jp/ca/admin

合格者データが入っている CD-ROMを、情報システム事務部(担当: 服部)が このマシン(shima

2. 学部 入試担当者の作業
   1. 生田地区の適当なコンピュータで、Webブラウザ(Netscape Navigator 3.x以上、または、Interne
   2. 以下の行をクリックする。

ｻｲﾄ 'shimauma.mind.meiji.ac.jp' はｸﾗｲｱﾝﾄ認証を要求しました.

以下が, ｻｲﾄの証明書です:

証明書: Meiji University
署名者: Meiji University
暗号化: Export ｸﾞﾚｰﾄﾞ (RC4-40 , 40 ﾋﾞｯﾄ秘密鍵)

証明書を選択して下さい: Hiroyuki HATTORI(3)'s Meiji University ICAP Experimental CA ID

- shimauma.mind.meiji.ac.jp は、インターネットからはアクセス不可能 です。(MINDアクセスレベル2です。
- Webサーバと Webブラウザの間の通信が「暗号化」されている間は、Webブラウザの画面左下の「鍵ア

client auth test Web-page

(文責) 情報システム管理課
服部 (*hhat@isc.meiji.ac.jp*)(内線 4439)

接続 ホスト shimauma.mind.meiji.ac.jp に接続しました。応答を待っています...

図 2 SSL を用いたユーザ認証の様子

このような仕組みを用いることによって、証明書を提示した者が本人であるかどうかの認証をネットワークを経由して行うことができる。

2.2 アクセス制御リスト

Web サーバでは、「アクセス制御リスト」に記述されているルールに従い、Web ページのアクセス制御を行っている。

ここでは、オープンソースな Web サーバプログラムである apache をベースに SSL 機能を付加した Web サーバ (apache+SSL) を例にあげる。

apache+SSL において、証明書を用いたアクセス制御を行う際、アクセス制御リストは以下のように記述する。

アクセス制御リスト (.htaccess)

```
AuthUserFile /opt/local/apache/etc/htpasswd
AuthName ByPassword
AuthType Basic

<Limit POST GET PUT>
  require valid-user
</Limit>
```

さらに、ユーザ情報を記載する htpasswd は以下のように記述する。

/opt/local/apache/etc/htpasswd

```
/C=JP/O=Meiji University/OU=Information Science Center/CN=hhat:xxj31ZMTZzkVA
/C=JP/O=WIDE Project/OU=Meiji Univ./CN=Hattori Hiroyuki:xxj31ZMTZzkVA
```

書式は、次の通りである。

/opt/local/apache/etc/htpasswd

```
[証明書の所有者名 (Subject )] : [パスワード ]
```

証明書の所有者名は、証明書の Subject フィールドの内容を記載する。パスワード部分の「xxj31ZMTZzkVA」は証明書を使ってアクセス制御する場合のおまじないの固定文字列で、ユーザに共通である。これは、Apache+SSL 固有の設定である。

2.3 アクセス制御リストのフォーマットの拡張

2.2で述べたアクセス制御リストはユーザ毎に記述する為に、ユーザ数が多い場合は大変な工数がかかる。

そこで、Apache+SSL に改良を加え、アクセス制御リストに正規表現を使えるようにした [3] [4]。

これにより、例えば「/C=JP/O=Meiji University/OU=Information Science Center」という Subject の証明書を持っているユーザならば、誰でもアクセスを許可する場合には、

/opt/local/apache/etc/htpasswd（正規表現

```
/C=JP/O=Meiji University/OU=Information Science Center/CN=.*:xxj31ZMTZzkVA
```

と一行で表現することができるようになった。

2.4 問題点

証明書を用いたユーザ認証を行う場合は、Webサーバ(認証する側)はユーザ(認証される側)から提示された証明書をチェックする。

チェック項目には、

1. 証明書の有効期限が切れていないか？
2. 証明書が偽造されたものでないかどうか？
3. 証明書がWebサーバの信頼するCAから発行されたものであるかどうか？

などがある。

この為、Webサーバ側では、ユーザの証明書を発行したCA自身の証明書を保持し、証明書が偽造されたものでないかどうかをチェックする仕組みになっている。

しかし、証明書のチェック項目は上記だけでは不完全である。

なぜならば、ユーザの秘密鍵が第三者に漏洩してしまった場合は、その者が正規のユーザになりすますことができてしまうからである。この場合、ユーザはCAを通して、ユーザの証明書を無効にする手続きが必要となる。

CAは、このような理由で無効にした証明書をCRL (Certificate Revocation Lists)にまとめ、定期的に公表する。

よって、証明書を用いてアクセス制限を行っているWebサーバは、CAから定期的に発行されるCRLを元に、アクセス制御リストを編集しなければならないが、このタイムラグがもとで、不正アクセス発生の可能性がでてくる。つまり、ユーザの秘密鍵を入手した第三者は、CRLに掲載される前ならば、あるいはWebサーバのアクセス制御リストが変更される前ならば、正規のユーザになりすまして不正にアクセスすることが可能となるからである。

これを解決する為には、WebサーバからCAに対して、リアルタイムに証明書の有効性確認を行えるような仕組みが必要となる。

3 証明書のリアルタイム検証機能の実現

前節で述べた、WebサーバからCAに対して、リアルタイムに証明書の有効性確認を行えるような仕組みを実現する為に、以下のシステムを開発した。

1. オンラインでの証明書の有効性確認が可能なCAの開発
2. リアルタイムでの証明書の有効性確認を行うWebサーバの開発

システムの全体像を、図3に示す。

3.1 オンラインでの証明書の有効性確認が可能なCAの開発

3.1.1 証明書発行局パッケージ - ICAP -

オンラインでの証明書有効性確認機能は、筆者らが開発した証明書発行局パッケージICAP [1]に対し、機能を追加することによって実現した。ICAPのシステム構成を図4に示す。

証明書有効性確認機能を実現するにあたり、HTTP(Hyper Text Transfer Protocol)をベースにした、次の機能を実現するプロトコルを定義した[1][5] [8] [9]。

---

[1] なお HTTP を用いた証明書検証などの問い合わせプロトコルは、現在、IETF (InternetEngineering Task Force) へ提案中である。

1. 証明書発行要求と応答 (certreq)
2. 証明書検索要求と応答 (lookupreq)
3. CA 証明書検索 (calookupreq)
4. CRL 検索要求と応答 (crlreq)
5. 証明書有効性確認要求と応答 (verifyreq)
6. 証明書更新要求と応答 (updatereq)
7. 証明書破棄要求と応答 (revokereq)

これらの機能を実装することによって、ネットワーク経由による証明書の有効性確認要求に対して、リアルタイムに応答することが可能になったばかりでなく、証明書の発行を複数の CA で分散して行う環境、つまり多段階 CA を構築した場合にも応答できるようになった [2]。

ICAP への問い合わせプロトコルの例として、以下に証明書の有効性確認 (verifyreq) のフォーマットと実行例をあげる。

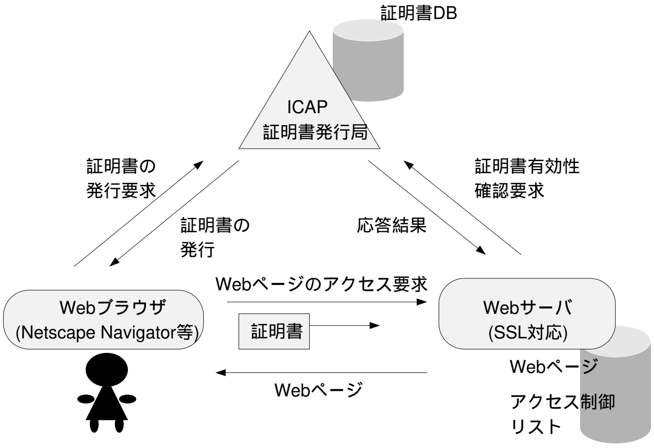


図 3 **ICAP と Web サーバとの連携イメージ**

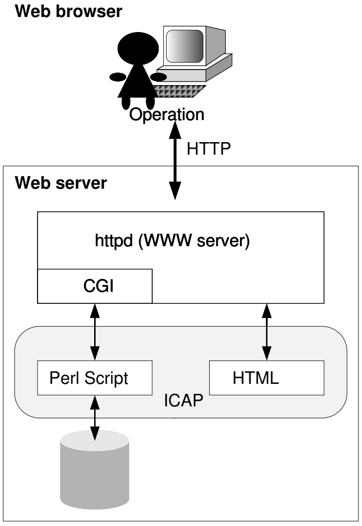


図 4 **ICAP のシステム構成**

証明書有効性確認機能 (verifyreq) のフォーマット

1. 要求フォーマット
    - HTTP の POST メソッドを利用する
    - パラメータは変数名=値の形式で渡す

```
POST /cgi-bin/verifyreq HTTP/1.0
Content-length:  (文字数)

 (パラメータ)
```

2. 応答フォーマット
    - 要求の種類
    - (要求に応じて定義される) 応答コード (statusCode) + 理由 (statusMessage)
    - その他のデータ (information)
3. パラメータ

```
====================================================
変数名          値
====================================================
cert            証明書の Base64 符号化形式
resptype        応答のタイプ (1 or 2)
                  1: 署名なし判定情報
                  2: 署名つき判定情報
====================================================
```

4. 応答
    - (正常時)

```
statusCode      statusMessage
-----------     -------------
200             "valid"
201             "invalid"
202             "revoked"
203             "expired"
204             "hold"

information
-----------
取り消し時刻 (statusCode = 202 の時のみ、UTCtime)
```

- (異常時)

```
statusCode statusCode の意味と説明
---------- -----------------------------------------------
303        "reject your request" any other reason why the request failed.
304        "service not avialable" the PA does not accept this request.

statusCode      statusMessage
----------      ------------------------------
303             "not certificate format"
303             "the URL not found"
303             "unknown responce type"
303             "unknown CA"
303             "AuthorityInfoAccess not included"
303             "signature verification failed"
304             "timeout"
304             "service not available"
```

証明書有効性確認機能 (verifyreq) の実行例

```
% telnet shimauma 80
POST /cgi-bin/verifyreq HTTP/1.0
Content-Length: 1085

resptype=1&cert=MIIDCjCCArSgAwIBAgIBATANBgkqhkiG9w0BAQQFADBmMQswCQYDVQQGEwJKUDEu
MCwGA1UEChMlTWVpamkgVW5pdmVyc2l0eSBJQ0FQIEV4cGVyaW1lbnRhbCBDQTEn
MCUGA1UECxMeSW5mb3JtYXRpb24gU3lzdGVtIE1hbmFnZW1lbnQgMB4XDTk5MDcw
MzAyMzIwNVoXDTk5MTIzMDAyMzIwNVowfTELMAkGA1UEBhMCSlAxLjAsBgNVBAoT
JU1laWppIFVuaXZlcnNpdHkgSUNBUCBFeHBlcmltZW50YWwgQ0ExGTAXBgNVBAMT
EEhpcm95dWtpIEhBVFRPUkkxIzAhBgkqhkiG9w0BCQEWFGhoYXRAaXNjLm1laWpp
LmFjLmpwMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALUmauJe14eYm%2bchzTvUS3SU
fja4cvW3wtX3UWFiwBcpE/2OEc15NqjWs5NaEHtX/CjLjU6znCQg3iEHkYlJnrcC
AwEAAaOCATQwggEwMA8GA1UdEwEBAAQFMAMBAQAwPQYDVR0gAQEABDMwMTAvBgkq
gwiBnF8LBAEwIjAgBgYrBgEFAwQWFmh0dHA6Ly93d3cuaWNhdC5vci5qcC8wgYIG
CyqDCIGcXwsDAR0CAQEABHAwbqA2hjRodHRwOi8vc2hpbWF1bWEubWluZC5tZWlq
aS5hYy5qcC9jZ2ktYmluL2NhbG9va3VwcmVxoTSGMmh0dHA6Ly9zaGltYXVtYS5t
aW5kLm1laWppLmFjLmpwL2NnaS1iaW4vdmVyaWZ5cmVxMEMGA1UdHwEBAAQ5MDcw
NaAzoDGGL2h0dHA6Ly9zaGltYXVtYS5taW5kLm1laWppLmFjLmpwL2NnaS1iaW4v
Y3JscmVxMBQGCWCGSAGG%2bEIBAQEBAAQEAwIAoDANBgkqhkiG9w0BAQQFAANBAJm5
nyI9bHobQWBXCMWQ%2bjGVPIQyOkFLninKc8Hel9bvaP9Jt09DZkE3t9ctREYM5LAS
QQR/3l/1KmG/Cyk57xa%3d

HTTP/1.1 200 OK
Date: Sat, 17 Jul 1999 01:22:36 GMT
Server: Apache/1.3.4 (Unix)
Connection: close
Content-Type: text/plain

verifyreq
200 valid
```

3.1.2 証明書の拡張フィールド

ユーザ証明書を Web サーバに問い合わせる為に、証明書の有効性に関する問い合わせ先、および CRL に関する問い合わせ先のアドレス情報を証明書に組み込んだ。これらの情報は、証明書の拡張フィールドを用いて記載した。

証明書の拡張フィールドの定義

```
<AuthorityInfoAccess>

  AuthorityInfoAccess OBJECT IDENTIFIER ::=
                      {ICAT applications(3) ds(1) 29 2}  (OID は ICAT 独自のもの)

  AuthorityInfoAccessSyntax  ::=  SEQUENCE  {
       authorityInfo     [0] SEQUENCE OF GeneralName OPTIONAL,
       certStatus        [1] SEQUENCE OF GeneralName OPTIONAL }

  GeneralName ::= CHOICE {
       otherName                       [0]     anotherName,
       rfc822Name                      [1]     IA5String,
       dNSName                         [2]     IA5String,
       x400Address                     [3]     ORAddress,
       directoryName                   [4]     Name,
       ediPartyName                    [5]     EDIPartyName,
       uniformResourceIdentifier       [6]     IA5String,
       iPAddress                       [7]     OCTET STRING,
       registeredID                    [8]     OBJECT IDENTIFIER }
```

3.2 リアルタイムでの証明書の有効性確認を行う Web サーバの開発

3.2.1 システム構成

図 5 に、リアルタイムでの証明書の有効性確認を行う Web サーバのシステム構成を示す。

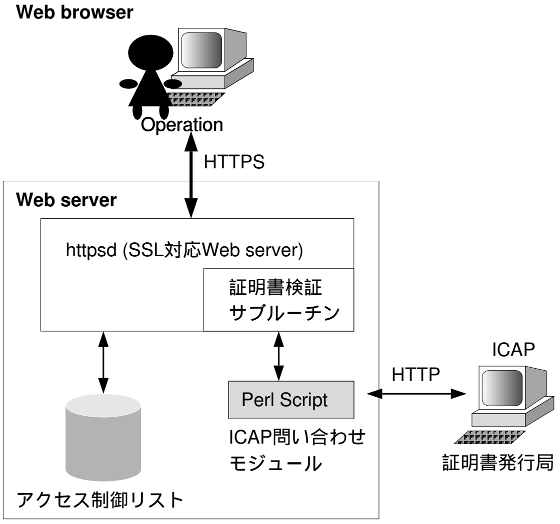


図 5 **Web サーバのシステム構成**

3.2.2 実装

ICAP への証明書有効性確認の問い合わせを行う機能は、以下のシステムをベースに開発した。

Web サーバ	Apache 1.3.6 + ssl_1.34
SSL	openssl-0.9.2b

図 5 において、ICAP 問い合わせモジュールは、Perl スクリプトで作成した。そして、Web サーバの証明書検証サブルーチン部分を修正し、ここから ICAP 問い合わせモジュールを呼び出すことによって、Web サーバから ICAP への問い合わせ機能を実現した。

3.3 Web サーバのオーバーヘッドに関する検証

ユーザ証明書の有効性確認をリアルタイムに行う処理には、どの程度の時間を要するのか、Web ページを表示するまでにどの程度のオーバーヘッドが発生するのかを、実験により検証した。

3.3.1 実験環境

- Web サーバ
  - S7/400Ui m270D (富士通製。Sun Microsystems 社 Sun Ultra5 相当品)
    - CPU: UltraSPARC IIi, 270MHz
    - Memory: 128MB
    - OS: Solaris 2.6
- ICAP サーバ
  - システム構成は Web サーバと同一。

- ネットワーク環境
  - Web サーバと ICAP サーバは、それぞれ 10Mbps で１台のイーサネットスイッチに接続している。実験中、Web サーバと ICAP サーバは他サーバからのトラフィックの影響を防ぐ為に、ネットワーク上の他のマシンからのアクセスは必要最低限のもの (DNS など) になるよう配慮した。

3.3.2 実験結果

実験は、Web サーバから ICAP サーバに対し、同一証明書を連続 20 回検証した時の実時間および、処理の内訳時間について調べた。

その結果、１件あたりの問い合わせに対する処理時間は次の通りとなった。

実時間	1.65 秒
(内訳)	
ネットワーク転送時間	0.0001 秒
ICAP 処理時間	1.60 秒
その他	0.0499 秒

このように検証要求に対する ICAP 内部での処理時間が、全体の処理時間の９０％以上を占めている。１件あたりの問い合わせに対する処理の実時間を短縮するには、ICAP 内部での処理時間を短縮するのが効果的である。

ICAP 内部で、証明書の有効性確認要求は、CGI 経由で起動された verifyreq モジュールが処理している。この verifyreq モジュールは、Perl スクリプトで記述されている。一般的に CGI からの Perl スクリプトの起動は、Web サーバにとって負担の重い処理であることが知られている。

よって、verifyreq モジュールをバイナリ化し、さらに CGI を利用せずに、もっと軽快に起動できるよう変更を加えれば、パフォーマンスは劇的に向上するものと思われる。

4 まとめ

本稿では、証明書を用いた安全なアクセス制御を実現する為に、ユーザの証明書をリアルタイムに検証することができるシステムを構築した。

3.3.2で述べた通り、一件当たりの証明書確認に 1.6 秒強を有してしまう。現段階では十分なパフォーマンスを得られているとは言いがたいが、これは ICAP 側のシステムを改良することによって、大幅に改善できると思われる。さらに、Web サーバ側での証明書のキャッシュなども有効であると思われる。

証明書の発行を複数の CA で分散して行う環境、つまり多段階 CA を構築した場合、CA 間の連携インフラストラクチャの構築が必須である。IETF では、さまざまな方式が提案されているが、決定的に普及している方式はまだ無い。

これには、Web サーバなど、認証する側において、CA との連携を実現する為のプログラムモジュールやライブラリの開発の遅れが、一つの原因と思われる。

筆者らが提案している Web ベースの問い合わせプロトコル普及の為には、本稿で Web サーバに搭載した証明書有効性確認モジュールなどを更に洗練し、サーバに容易に組み込むことができるプログラムライブラリとすることが必要になっており、これが今後の重要な課題であると考えている。

A 謝辞

本研究にあたり、有益な意見を頂いた認証実用化実験協議会 (ICAT)、広域認証技術タスクフォースのメンバーに感謝いたします。

B 参考文献

参考文献

[1] 服部裕之、櫻井三子、小林良至、菊池浩明：“オンライン証明書発行局パッケージ (ICAP) の実装と評価”, 1997 年暗号と情報セキュリティ・シンポジウム,SCIS’97-8C,(1997)

[2] 櫻井三子、服部裕之、小林良至、菊池浩明：“証明書発行局間の証明書情報共有機構の設計”, 1997 年暗号と情報セキュリティ・シンポジウム,SCIS’97-8D,(1997)

[3] 服部裕之：“Apache+SSLeay を用いた、証明書ベースによる認証機能の活用”, 認証実用化実験協議会 (ICAT） 広域認証技術研究タスクフォース 配布資料,(1997) (available ftom http://www.isc.meiji.ac.jp/h̃hat/report/client_auth.txt)

[4] 服部裕之：“SSLeay と Apache によるセキュアサーバの構築について”, 認証実用化実験協議会 (ICAT)、’97 定例研究会, (1997) (available from http://www.isc.meiji.ac.jp/h̃hat/ppt/teirei97_1)

[5] H.Kikuchi, M.Sakurai, Y.Sameshima, H.Hattori : “*Internet Public Key Infrastructure:Web-based Certificate and CRL Repository*” , Internet Draft, (1997) (available from ftp://rs.internic.net/internet-drafts/ draft-kikuchi-web-cert-repository-00.txt)

[6] 櫻井三子、服部裕之：“moCA 実験報告”, WIDE November’97 研究会, (1997)

[7] 櫻井三子、服部裕之：“WIDE プロジェクトにおける CA 運用実験と考察”, 1998 年暗号と情報セキュリティ・シンポジウム,SCIS’98-3.3B,(1998)

[8] M.Sakurai, H.Kikuchi, H.Hattori, Y.Sameshima, H.Kumagai : “*Web-based Integrated CA services Protocol, ICAP*” Internet Draft, (1998) (available from ftp://rs.internic.net/internet-drafts/ draft-sakurai-pkix-icap-00.txt)

[9] M.Sakurai, H.Kikuchi, H.Hattori, Y.Sameshima, H.Kumagai : “*Web-based Integrated CA services Protocol, ICAP*” Internet Draft, (1999) (available from ftp://rs.internic.net/internet-drafts/ draft-sakurai-pkix-icap-01.txt)

[10] 明治大学実験 CA：
http://www.isc.meiji.ac.jp/ca/index.html