Chapter 2: Foundation - Identifiers and Protocols (cont.)

MD5 – message digest - RFC1321

"The MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner [...]"

The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. The MD5 algorithm is an extension of the MD4 message-digest algorithm. MD5 is slightly slower than MD4, but is more "conservative" in design. [...] The MD5 algorithm is being placed in the public domain for review and possible adoption as a standard." Source: RFC 1321.

HTTP/1.1, Authentication

- improved authentication
  - Digest Access Authentication
  - Basic Authentication still possible and supported
- procedure
  - response with status-code: 401 Unauthorized and header
    WWW-Authenticate: Digest realm="...", nonce="..."
  - user is asked to provide password
  - request to the resource with additional header fields
    Authorization: Digest realm="...", username="...", response="..."
    response = <MD5(<passwd>:<nonce>:...)>
  - <passwd>:<nonce> is MD5 coded, reverse is not possible
  - server calculated also MD5 of <passwd>:<nonce>:...
    if the response matches the servers result the document is transferred
- password is not sent!
- problem: initial exchange of passwords
Digest Access Authentication I

GET / HTTP/1.1
Accept: */*
Host: 129.13.170.61
User-Agent: amaya/V2.2 libwww/5.2.8
Connection: Keep-Alive

HTTP/1.1 401 Authorization Required
Date: Tue, 02 Nov 1999 16:05:36 GMT
Server: mod_perl/1.18 Apache/1.3.4 (Unix) (SuSE/Linux) PHP/3.0.7
WWW-Authenticate: Digest realm="teco61pc", nonce="941558736"
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<!-- HTML content -->

Digest Access Authentication II

GET / HTTP/1.1
Accept: */*
Authorization: Digest username="albrecht", realm="teco61pc",
nonce="941558736", uri="http://129.13.170.61/"
response=06430f9c0dd1473e856206a4ab040f6d
Host: 129.13.170.61
User-Agent: amaya/V2.2 libwww/5.2.8
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 02 Nov 1999 16:05:41 GMT
Server: mod_perl/1.18 Apache/1.3.4 (Unix) (SuSE/Linux) PHP/3.0.7
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html

Digest Access Authentication III

Client

GET /dir/index.html
Authorization: Digest username="albrecht", realm="teco",
nonce="12123343", response="e966c932a9242554e42c8ee200ce7f6"
User-Agent: amaya/V2.2 libwww/5.2.8
Connection: Keep-Alive

Server

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="teco", nonce="12123343"

Check MD5

Encryption

- authentication has nothing to do with encryption of the data transmitted!
- HTTPS = HTTP over SSL (secure socket layer)
- S-HTTP = secure HTTP [Rescorla and Schiffman 1997]
### Chunked Encoding
- Chunked encoding for dynamic resources
  - when size of the document is not known before (e.g. database access)
  - sequence of \(<\text{Chunk Size}, \text{Chunk Data}>\) pairs
  - header field: \texttt{Transfer-Encoding}
- Example

```
HTTP/1.1 200 OK
Date: Tue, 26 Oct 1999 16:56:37 GMT
Server: Apache/1.2.1
Transfer-Encoding: chunked
Content-Type: text/html

<html><head>
<title>Telecooperation Office (TecO)</title>
...</td width=40> &nbsp;
</body></html>
```

### Content Negotiation
- Language specific resources
- Resources of different quality
- Resources using different coding
- Header fields: e.g.
  - Accept,
  - Accept-Charset,
  - Accept-Language,
  - Accept-Encoding, ...
- Different concepts: server-driven, agent-driven, transparent
- Selection by
  - Server – based on the request the appropriate resource is delivered
  - User/browser – server sends a list of choices:
    - status-code: 300 (multiple choices)
- Knowing the software (browser, operating system)
- Appropriate resources can be provided,
  \texttt{User-Agent: Mozilla/4.6 [en] (WinNT; I)}

### Server Driven - Content Negotiation I
- Based on the request the appropriate resource is delivered
- Implementation examples
  - CGI/ASP/PHP/SSI evaluate incoming request and decide then what the appropriate resource is that will then be delivered.
    ```perl
    if ($ENV{HTTP_USER_AGENT} =~ / mozilla/i) {
      print "Content-Type: text/html\n\n"
      open(FILE1, $file1) || die "file1: $! \n"
      while (<FILE1>) { print; }
      close FILE1;
    } else { if ($ENV{HTTP_USER_AGENT} =~ / wapide/i) {
      print "Content-Type: text/vnd.wap.wml\n\n"
      open(FILE2, $file2) || die "file2: $! \n",
      while (<FILE2>) { print; }
      close FILE2;
    }
    
    # may be another type ...
    print "Content-Type: text/vnd.wap.wml\n\n"
    open(FILE2, $file2) || die "file2: $! \n",
    while (<FILE2>) { print; }
    close FILE2;
    }
    ```
- WML und HTML using the same URL on Apache
  ```apache
  RewriteCond %{HTTP_ACCEPT} text/vnd.wap.wml
  RewriteRule /$ /index.wml [T=text/vnd.wap.wml]
  ```

---

### Server Driven - Content Negotiation II
Further Header Fields I

- **User-Agent**
  - Information about the browser and usually also about the operating system the user is using.
  - Can be used to provide optimized pages for the platform.
  - Placing different adverts for different platforms (even different prices).
  - For statistical use (knowing the customer’s platform the page can be optimized for this platform).
  - E.g.:
    - User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)
    - User-Agent: Mozilla/4.6 [en] (WinNT; I)
    - User-Agent: amaya/2.2 libwww/5.2.8

- **Server**
  - Information about the server software, sometimes also about the operating system.
  - Exact information could be a risk; hackers know then where/how to attack.
  - E.g.:
    - Server: Apache/1.3.10 (Unix) ApacheJServ/1.0 PHP/3.0.6
    - Server: Microsoft-IIS/5.0

Further Header Fields II

- **Referer**
  - Information about the last page visited.
  - URL of the page on which the link to the current page was used to get to the current page.
  - Is only transmitted when a link was clicked, not when a new link is manually typed in.
  - Usage/purpose:
    - Make sure that a service (page, script, ...) is only called after visiting a certain page (URL).
    - If the user comes from a different path (other page with a link to this service), the system can use the referer to determine this and deny the function (e.g. free SMS service).
    - Statistical analysis, where are users coming from? What adverts are successful?
    - Search engines transmit with this mechanism the query the user put in; this can then be used to optimize page that the users will find them.
  - E.g.:
    - Referer: http://www.teco.edu/lehre/webe/beispiele.htm
    - Referer: http://www.altavista.com/cgi-bin/query?q=%2Bperl+%2Bhttp...

Server Initiated Communication

- **HTTP is a Request-Reply Protocol**
  - Requests are initiated from the client.
  - A server can not initiate the communication to a client.

- **“Workarounds”**
  - Client pulls periodically for information.
    - Using a standard browser, e.g.:
      - Reload a page every 60 seconds:
        - `<META HTTP-EQUIV= "Refresh" content="60">`
      - Reload a new URL:
        - `<META HTTP-EQUIV= "Refresh" content="0; URL=http://www.teco.edu/">`
    - Problem: Scalability.
    - Specific clients using different protocols, e.g.:

Cookies

- **HTTP is stateless**
  - Two consecutive requests are not connected at protocol level.

- **Using Cookies the state (or and ID that represents the state) is stored on the client side.**
  - Introduced by Netscape.

- **Protocol primitives**
  - **Set-Cookie**
    - The request from the server to client to store a cookie, included in the reply header.
  - **Cookie**
    - If a cookie is stored for the current domain and path of the request this is sent to the server, included in the request header.

- **Used to implement sessions (e.g. shopping cart)**
  - CGI, ASP, etc.
  - See chapter 6.
  - Libraries, e.g.:
Set-Cookie – Syntax (RFC 2109)

```plaintext
set-cookie = "Set-Cookie:" cookie

cookie = NAME "=" VALUE *(";" cookie-av)

cookie-av = "Comment" "=" value
 | "domain" "=" value
 | "Max-Age" "=" value
 | "path" "=" value
 | "Secure"
 | "Version" "=" 1*DIGIT
```

Cookie – Syntax (RFC 2109)

```plaintext
cookie = "Cookie:" cookie-version
1*(((";" | ",") cookie-value)

cookie-value = NAME "=" VALUE [";" path] [";" domain]

cookie-version = "$Version" "=" value
```

Cookies – Example I

```plaintext
GET / HTTP/1.1
Accept: */*
Host: www.hotbot.com

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Wed, 03 Nov 1999 02:57:09 GMT
Set-Cookie: p_uniqid=48BpFe5tJWdl+7QaB;
            expires=Fri, 21-Dec-2012 08:00:00 GMT;
            domain=.hotbot.com; path=/
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 15982

<html>
<head><title>HotBot</title>
...
```

Cookies – Example II

```plaintext
GET / HTTP/1.1
Accept: */*
Host: www.hotbot.com

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Wed, 03 Nov 1999 02:57:09 GMT
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 15982

<html>
<head><title>HotBot</title>
...
```
Preserving State in Communication

Client

Server/Application

- Request
- Response + Set-Cookie
- Request + Cookie
- Response
- Request + Cookie
- Response

use/update Cookie

create Cookie

New Challenges after HTTP/1.1?

- Scenario 1: access on a very large scale, e.g. Advertisement: www.tickets-4-free.com during a major sport event
- Scenario 2: controlling and operating a power plant over the WWW using HTTP
- Scenario 3: shared editing of multimedia documents
- Scenario 4: off-line access to documents (e.g. Pre-fetch, advanced caching)
- Scenario 5: Copyright and pay-per-view mechanisms
- Scenario 6: documents that consist of parts that are broadcasted and parts that are requested (e.g. Vision of WebTV)

HTTP/1.1

- RFC 2616 – HTTP/1.1 Draft Standard (July 1999)
- RFC 2617 – HTTP Authentication (June 1999)

Basic and Digest Access Authentication

Implementations (examples and libraries):
- Web server: http://www.w3.org/Jigsaw/
- Browser/Editor: http://www.w3.org/Amaya/
- Libwww (library) http://www.w3.org/Library/

What are the achievements of HTTP/1.1?
- HTTP/1.1 works better and more efficient with TCP
- a clear Caching Semantic is introduced
- wasting of IP-Addresses is reduced

Performance und QoS

- HTTP using TCP over IP
  → best effort

- speed
- bandwidth
- scalability
- delay, jitter
- reliability
- replication
Shared Editing

- locks on objects and on parts of objects
  - read
  - write
  - information on locks
- distribution
  - link management
  - management of resources
- versioning, revision control
  - replication
- delete/rename/move of objects
  - dependencies
- see: [http://www.w3journal.com/4/s3.whitehead.html](http://www.w3journal.com/4/s3.whitehead.html)

Further Challenges

- real-time behavior
- state management
- asynchronous messages
- Information about cache access
- compression
- ...

PEP (Protocol Extension Protocol)

- So far: HTTP can be extended using new header fields, but:
  - no agreement about the structure of extensions
  - no registration of extensions
  - no mechanisms to group header fields that belong together
  - no defined behavior on unknown header
- Ideas for a solution (and further questions):
  - new Header fields: Protocol, Accept-Protocol
  - who should understand the protocol? server, all stations on the route, next station?
  - how should be dealt with multiple extensions or inconsistent extensions?
  - negotiation of the protocol (must, optional, refused)
- see: [http://www.w3.org/Protocols/PEP/](http://www.w3.org/Protocols/PEP/)

HTTP-NG

- HTTP Next Generation [Janssen et al. 1998]
  - new protocol architecture
  - asynchronous requests and responses including state
  - multiplexing on transport level
  - not only over TCP/IP, also other protocols, e.g. ATM
  - security framework
HTTP NG

References Chapter 2 (I)

- Berners-Lee, T., Fielding, R.T., and Nielsen, H.F.; 1996. Hypertext Transfer Protocol - HTTP/1.0. Internet informational RFC1945

References Chapter 2 (II)

- W3C; Diverse Dokumente über Protokolle. http://www.w3.org/Protocols/ oder http://www.w3.org/pub/WWW/Protocols/