# Web Auth

Abram Hindle
abram.hindle@ualberta.ca
Department of Computing Science
University of Alberta
http://softwareprocess.es/

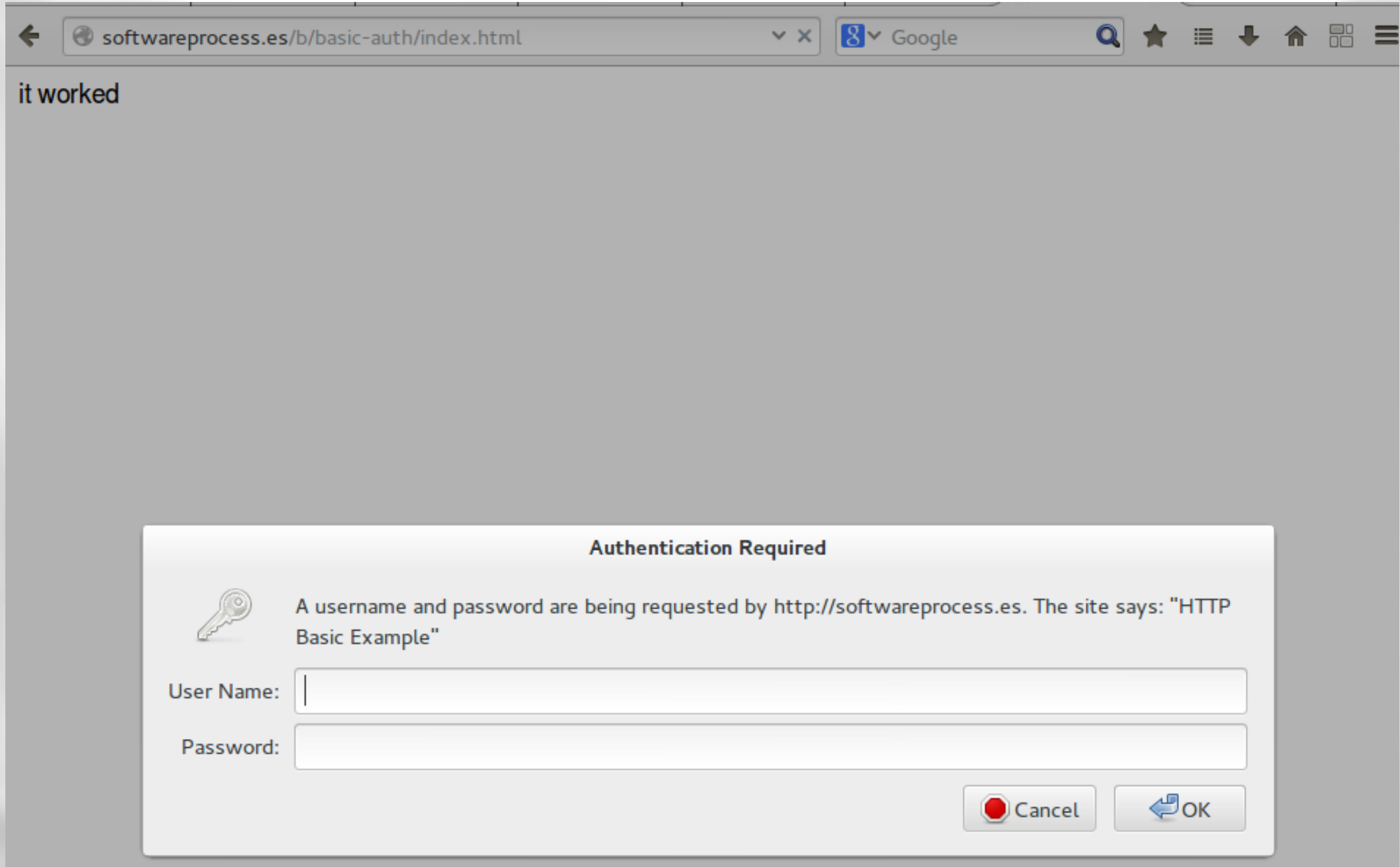
# Web Authentication

- The methods of authenticating users according to the HTTP spec.
- Somewhat different from alternative auth means such as secure tokens, cookies, and sessions, but often used together.
- Since software is on the web, we need to apply the same protections we apply to software.

# HTTP Authentication: Basic and Digest Access Authentication

- Defined in http://www.ietf.org/rfc/rfc2617.txt by Franks et al. 1999
- 2 forms of auth
  - Basic
    - User name and password sent as “clear text”
    - Unsafe unless you use SSL/TLS (HTTPS)
  - Digest
    - Using cryptographic hashes and shared secrets to authenticate. Slightly safer, can still get hijacked.
- Hint: Use HTTPS all of the time ;-)

# HTTP Basic


softwareprocess.es/b/basic-auth/index.html
Google
it worked
Authentication Required
A username and password are being requested by http://softwareprocess.es. The site says: "HTTP Basic Example"
User Name:
Password:
Cancel
OK

# HTTP Basic

- Easiest and “stateless”
- User accesses a resource and a 401 unauthorized is returned but a WWW-Authenticate header is sent.
  - WWW-Authenticate: Basic realm="Name of your realm"
    - Realm is what you are authenticating for
  - User-agent responds with:
    - Authorization: Basic aGluZGxlMTpwYXNzd29yZDE=
      - Base64(userid + “:” + password)
      - Base64(“hindle1:password1”) === “aGluZGxlMTpwYXNzd29yZDE=”

# HTTP Basic Assumptions

- Client should be access all paths “at or deeper than the depth of the last symbolic element in the parth field”
  - So all files and subdirs within whatever you authenticated with.
- You can send authorization as much as you like.

# HTTP Basic

- HTTP Basic over HTTP
  - Unencrypted
  - Insecure
  - Stealable
  - Unsafe
  - Proxyable
  - Stateless

```
hindle1@piggy:~$ curl -v http://softwareprocess.es/b/basic-auth/index.html
* Adding handle: conn: 0x2249b70
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x2249b70) send_pipe: 1, recv_pipe: 0
* About to connect() to softwareprocess.es port 80 (#0)
*   Trying 75.119.223.206...
* Connected to softwareprocess.es (75.119.223.206) port 80 (#0)
> GET /b/basic-auth/index.html HTTP/1.1
> User-Agent: curl/7.32.0
> Host: softwareprocess.es
> Accept: */*
>
< HTTP/1.1 401 Authorization Required
< Date: Mon, 24 Mar 2014 04:35:20 GMT
* Server Apache is not blacklisted
< Server: Apache
< WWW-Authenticate: Basic realm="HTTP Basic Example"
< Last-Modified: Tue, 26 Jun 2012 16:34:47 GMT
< ETag: "0-4c362ad3537c0"
< Accept-Ranges: bytes
< Content-Length: 0
< Vary: Accept-Encoding
< Content-Type: text/html; charset=utf-8
<
```

```
hindle1@piggy:~$ curl -v -u username:password1 http://softwareprocess.es/b/basic-auth/in
* Adding handle: conn: 0x12c1bd0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x12c1bd0) send_pipe: 1, recv_pipe: 0
* About to connect() to softwareprocess.es port 80 (#0)
*   Trying 75.119.223.206...
* Connected to softwareprocess.es (75.119.223.206) port 80 (#0)
* Server auth using Basic with user 'username'
> GET /b/basic-auth/index.html HTTP/1.1
> Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQx
> User-Agent: curl/7.32.0
> Host: softwareprocess.es
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 24 Mar 2014 04:36:52 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Mon, 24 Mar 2014 04:31:15 GMT
< ETag: "a-4f552b4c850af"
< Accept-Ranges: bytes
< Content-Length: 10
< Vary: Accept-Encoding
< Content-Type: text/html; charset=utf-8
<
it worked
```

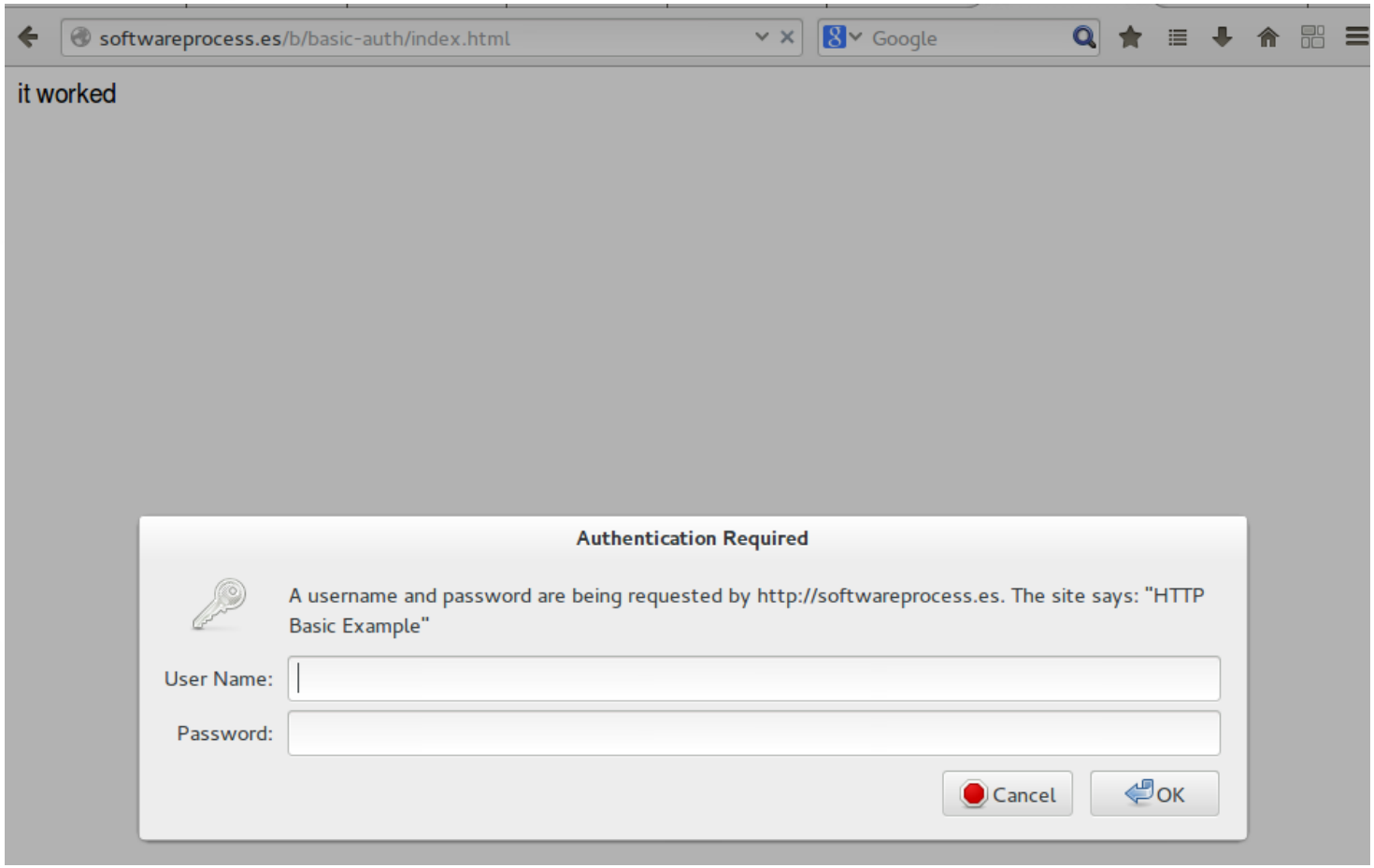
# HTTP Basic

```
> Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQx
```

```
hindle1@piggy:~$ base64 -d
dXNlcm5hbWU6cGFzc3dvcmQx
username:password1
```

So no encryption there!

# HTTP Basic


softwareprocess.es/b/basic-auth/index.html
Google
it worked
Authentication Required
A username and password are being requested by http://softwareprocess.es. The site says: "HTTP Basic Example"
User Name:
Password:
Cancel
OK

# HTTP Digest Authentication

- “Users and implementors should be aware that this protocol is not as secure as Kerberos, and not as secure as any client-side private-key scheme. Nevertheless it is better than nothing, better than what is commonly used with telnet and ftp, and better than Basic authentication.”

  -- Franks et al. 1999
  http://www.ietf.org/rfc/rfc2617.txt

# HTTP Digest Authentication

- Avoids sends a plaintext username and password across the wire
- All content traded could be listened on.
- No encryption, but a hashing and shared secret scheme.
- Can attempt to avoid replay attacks
  - Replay attack, whereby repeating a recorded message you fake being authenticated.
  - You copy one of my “going to lunch emails” and you send it to my colleague to trick my colleague to leave their office.

# HTTP Digest Run Down

- In general we form a bunch of strings using colons:
  - We hash them
  - We share them
  - Some of these strings are secrets

# HTTP Digest Run Down

- First the client requests data
- Then the server responds with a 401 and WWW-Authenticate: Digest ...args...
- Then the client repeats the request with a Authorization: header
- Then the server echos back much of that information and returns the appropriate content and status

```
hindle1@piggy:~$ curl -v --digest -u username:password1 \
     http://softwareprocess.es/b/digest-auth/index.html
* Connected to softwareprocess.es (75.119.223.206) port 80 (#0)
> GET /b/digest-auth/index.html HTTP/1.1
> User-Agent: curl/7.32.0
> Host: softwareprocess.es
> Accept: */*
>
< HTTP/1.1 401 Authorization Required
< Date: Mon, 24 Mar 2014 06:11:37 GMT
* Server Apache is not blacklisted
< Server: Apache
< WWW-Authenticate: Digest realm="HTTP Digest Example",
  nonce="nb6xG1T1BAA=d3ef815a59221504475406693386f990b8a7a3a4",
   algorithm=MD5,
   domain="/b/digest-auth",
   qop="auth"
< Last-Modified: Tue, 26 Jun 2012 16:34:47 GMT
< ETag: "0-4c362ad3537c0"
< Accept-Ranges: bytes
< Content-Length: 0
< Vary: Accept-Encoding
< Content-Type: text/html; charset=utf-8
<
* Connection #0 to host softwareprocess.es left intact
```

# Re-GET with Digest Auth

```
> GET /b/digest-auth/index.html HTTP/1.1
> Authorization: Digest username="username",
    realm="HTTP Digest Example",
    nonce="nb6xG1T1BAA=d3ef815a592215044754066933 86f990b8a7a3a4",
    uri="/b/digest-auth/index.html",
    cnonce="MjY3MGQyNWU4M2E0ZDFmMjAwMTJmMmVjMDAwZTgyN2I=",
    nc=00000001,
    qop=auth,
    response="523b3ab0a9d25185318b5d3cc9c634b5",
    algorithm="MD5"
> User-Agent: curl/7.32.0
> Host: softwareprocess.es
> Accept: */*
>
```

# Response to successful Digest

```
< HTTP/1.1 200 OK
< Date: Mon, 24 Mar 2014 06:11:37 GMT
* Server Apache is not blacklisted
< Server: Apache
< Authentication-Info:
  rspauth="c313b8fea0ac15efd27075a53d31994f",
  cnonce="MjY3MGQyNWU4M2E0ZDFmMjAwMTJmMmVjMDAwZTgyN2I=",
  nc=00000001,
  qop=auth
< Last-Modified: Mon, 24 Mar 2014 06:02:07 GMT
< ETag: "a-4f553f9b6078e"
< Accept-Ranges: bytes
< Content-Length: 10
< Vary: Accept-Encoding
< Content-Type: text/html; charset=utf-8
<
it worked
```

# HTTP Digest: Request Digest

- Request-digest for non-qop:

  request-digest = “\"” + KD( H(A1), nonce-value + ":" + H(A2) ) + “\"”
- H is often MD5
- KD(x,y) = MD5(x + “:” + y)
- A1 = username-value + ":" + realm-value ":" passwd
- A2 = “:” + digest-uri (authorization header request)

# HTTP Digest: Request Digest

- Request-digest for qop:

  request-digest = “\"” + KD( H(A1), “:”.join([nonce, nc, cnonce, qop,H(A2)] ) + “\"”
- H is often MD5
- KD(x,y) = MD5(x + “:” + y)
- A1 = username-value + ":" + realm-value ":" passwd
- A2 = Method “:” + digest-uri (authorization header request)

# HTTP Digest: response digest

- Sub header rspauth
- response-digest for qop (same as request digest):

  response-digest  = “\"” + KD( H(A1), “:”.join([nonce, nc, cnonce, qop,H(A2)] ) + “\"”
- H is often MD5
- KD(x,y) = MD5(x + “:” + y)
- A1 = username-value + ":" + realm-value ":" passwd
- A2 = Method “:” + digest-uri (authorization header request)
- EXCEPT it is hex encoded

# Let's do it by hand

```
WWW-Authenticate: Digest realm="HTTP Digest Example",

nonce="nb6xG1T1BAA=d3ef815a592215044754066933 86f990b8a7a3a4
", algorithm=MD5, domain="/b/digest-auth", qop="auth"
  def md5(v):
      m = hashlib.md5()
      m.update(v)
      return m.hexdigest()

  username = "username"
  passwd = "password1"
  realm = "HTTP Digest Example"
  a1 = ":".join([username,realm,passwd])
  a2 = ":".join(["GET","/b/digest-auth/index.html"])
  method="GET"
  uri="/b/digest-auth/index.html"
  qop="auth"
  nonce = "nb6xG1T1BAA=d3ef815a592215044754066933 86f990b8a7a3a4"
  nc="00000001"
  cnonce="MjY3MGQyNWU4M2E0ZDFmMjAwMTJmMmVjMDAwZTgyN2I="
  md5(md5(a1) + ":" + ":".join([nonce, nc, cnonce, qop, md5(a2)]))
  # OUT: '523b3ab0a9d25185318b5d3cc9c634b5'
```

# HTTP Digest Authentication

- Can be man in the middled
  - Someone could change HTTP headers on you and latch onto your authentication
- Offers no confidentiality
- Performance issues:
  - Each nonce update requires a reauth
  - Can't send a pre-auth'd request without chatting first.

# HTTP Digest Authentication

- Nonces
  - Need to be generated and changed often
  - A stale nonce is a broken system
  - Easy to break

# Authentication and REST?

- Digest will require a lot of hand shaking all over the place
- Basic is far simpler
- Basic works fine over HTTPS
  - Other alternatives
    - OpenID
    - Oauth
    - Cookies and Sessions but you might have strip authentication along the way to make it more restful
    - Tokens

# Authorization header

- In Oauth and Oauth2 the authorization header is overriden with a token:
  - Authorization: token OAUTH-TOKEN
  - Reusing HTTP infrastructure is probably a good idea
- Authorization token allows things to stay RESTful

# Resources

- RFC 2617 HTTP Authentication: Basic and Digest Access Authentication http://www.ietf.org/rfc/rfc2617.txt
- Github V3 Oauth
  - https://developer.github.com/v3/oauth/
- An HTTP Digest example
  - http://en.wikipedia.org/wiki/Digest_access_authentication#HTTP_digest_authentication_considerations