Open Source Software Project Development

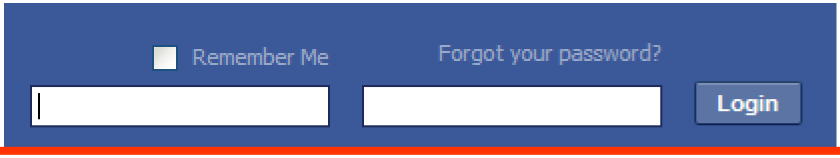
Dr. T.Y. Wong

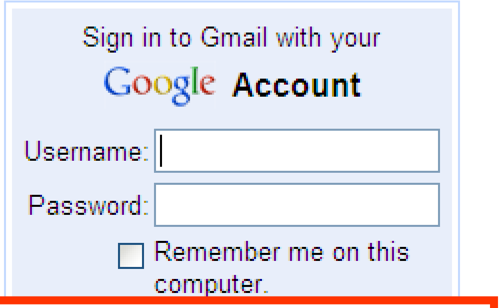
Weeks 4 – 6

Login Mechanism and Session Management

- be a good gate keeper...

Most web-systems start with…





Sign in to Gmail with your Google Account

Username:

Password:

☐ Remember me on this computer.

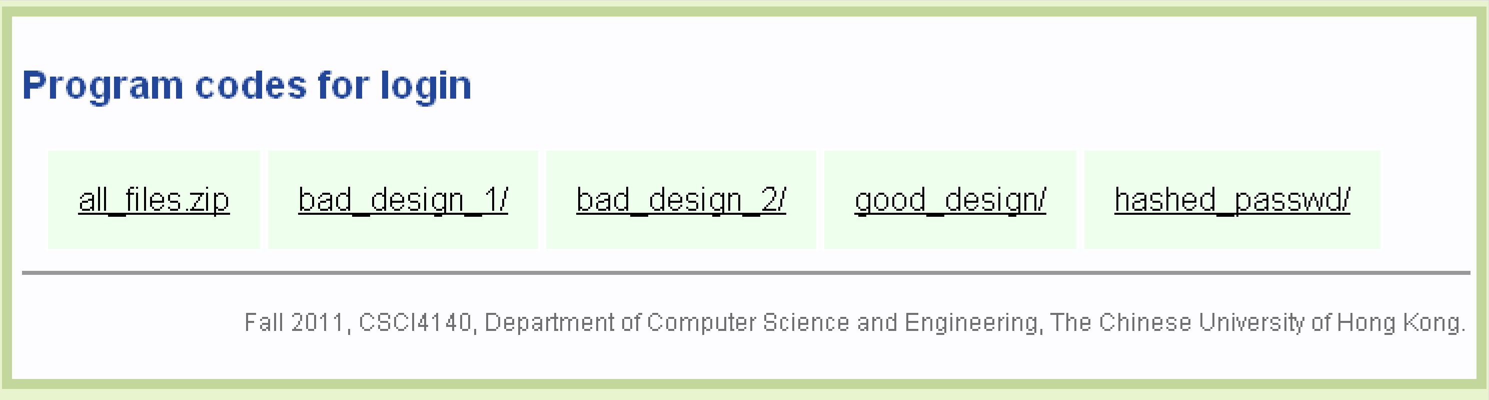
The Question is:

How to Realize the Login Process?

密碼		忘記密碼
安全提問	無安全提問	
回答		如果您設置了安全提問，請回答正確的答案
登錄有效期	○ 永久 ● 一個月 ○ 一天 ○ 一小時 ○ 瀏覽器進程	
隱身登錄	- 使用默認 -	
界面風格	- 使用默認 -	
	提交	

Things to take care of…

- Mechanism
  - Using web server’s capabilities?
  - Using homebrew CGI programs?
- User-password management?
  - Depending on which mechanism you’re using.
- Login session management
  - What is a session?
  - Why do we need a session?
  - How to maintain a session?



Program codes for login

all_files.zip | bad_design_1/ | bad_design_2/ | good_design/ | hashed_passwd/

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

http://demo4140-tywong.rhcloud.com/03_login

Login Using… Tailor-made CGI Program(s).

Using CGI programs?

- First,
  - Why not depending on the web server?

HTTP Authentication	
Pros	Cons
Password management is easy.	Not secure…or the secure way is not wildly adopted…
Session management is easy.	No varieties. E.g., password retention and revocation, logging.

Cons of HTTP authentication does not automatically convert to the pros of CGI programs.

Using CGI programs…

- Some facts…

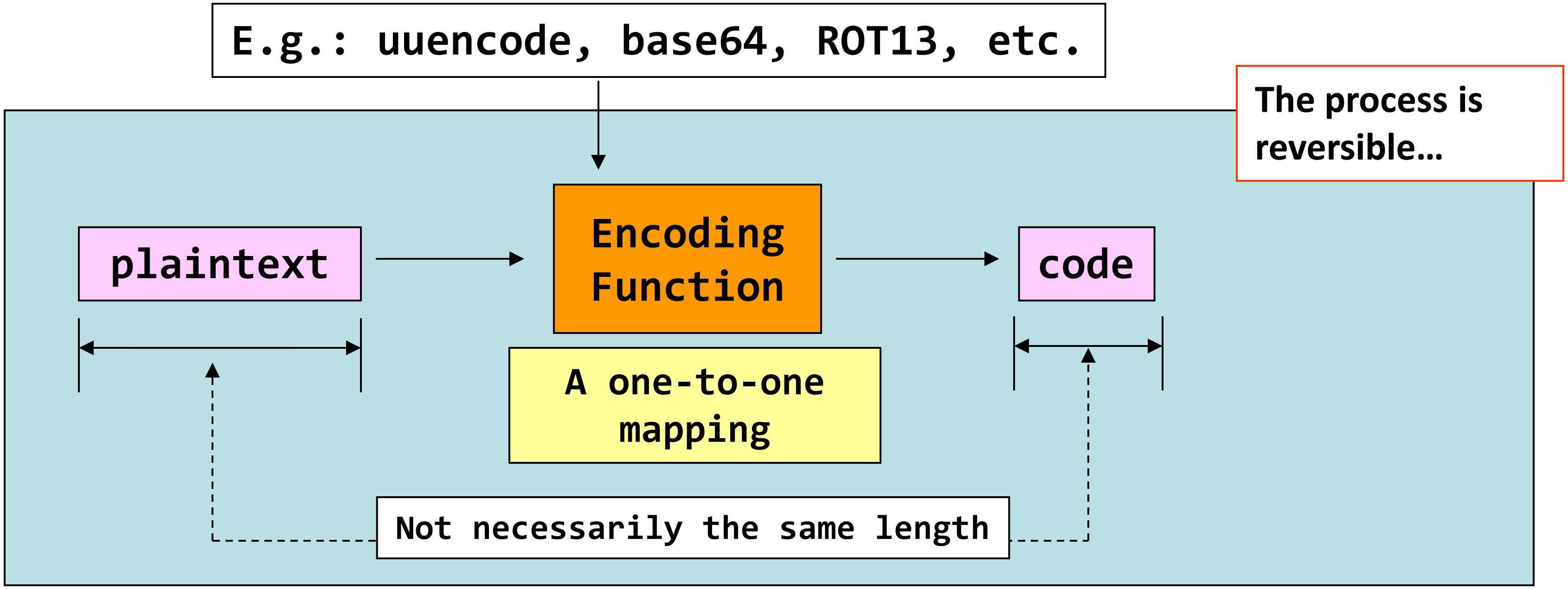
Using CGI programs…
- can only be as secure as using HTTP authentication.
- requires you to implement the password management.
- requires you to implement the session management.
- can give you a lot of flexibility in managing session and password.

Sidetrack

(in case you don’t know)

Encoding, Hashing, & Encryption

Sidetrack: Encoding


E.g.: uuencode, base64, ROT13, etc.
The process is reversible…
plaintext
Encoding Function
code
A one-to-one mapping
Not necessarily the same length

A encoding function is to **produce a reversible code**.

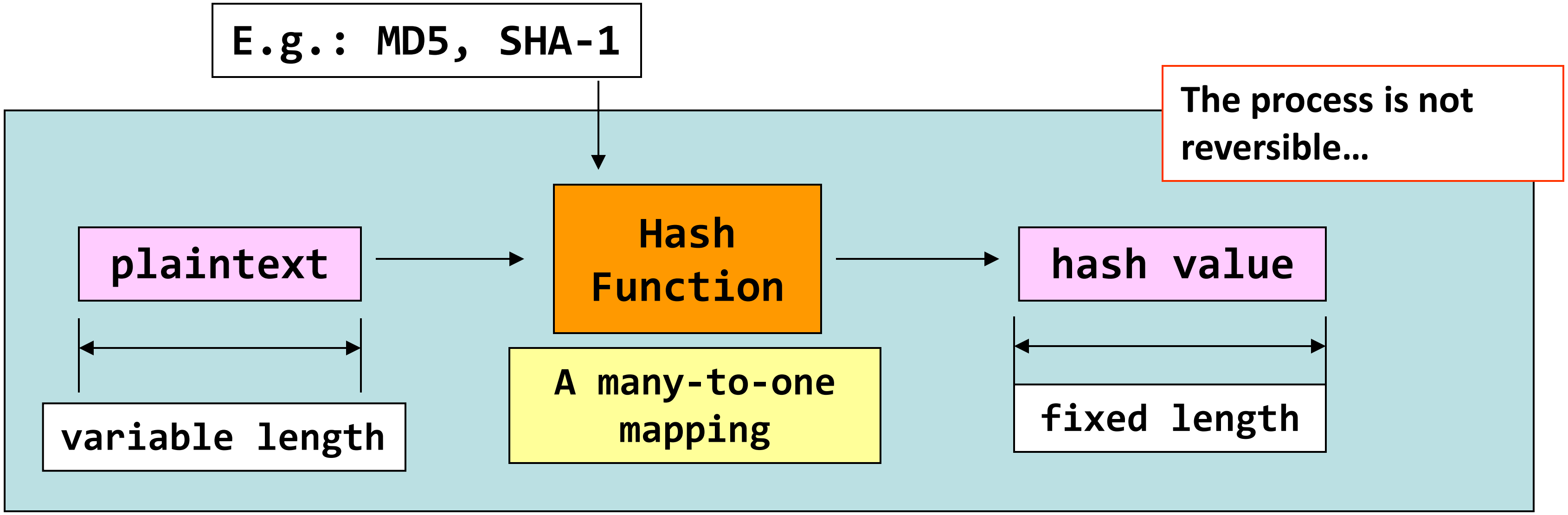
The propose is to either:
(1) mess things up; or
(2) convert the binary file to ASCII format.

Just to let you know:

base64 is to map every 64 bits from the input stream into a coded alphabet.

See wikipedia for details.

Sidetrack: Hashing



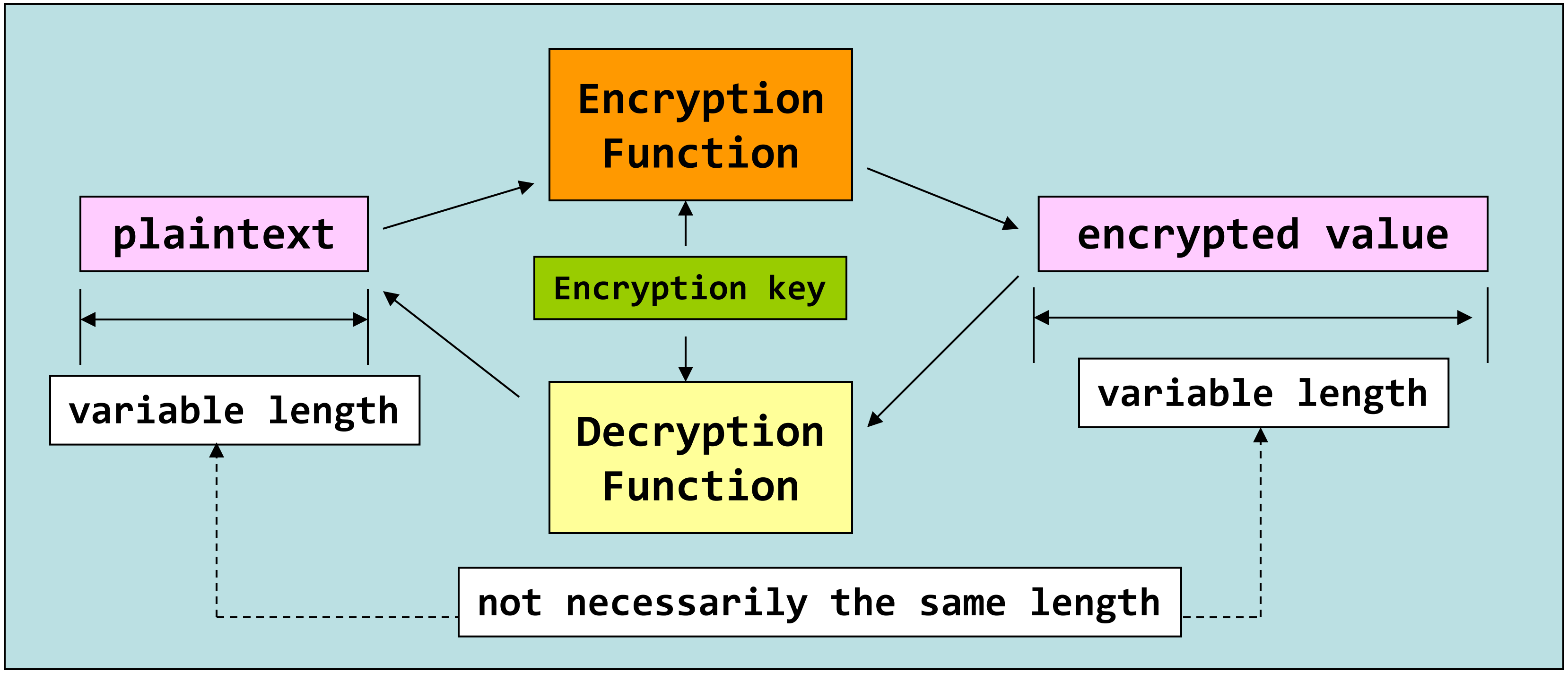
A hash function is to produce a **kind-of unique** representation of the input.

There are chances that two different input produces the same hashed value, but the chance is extremely small…

Just to let you know:

MD5 and SHA-1 are cracked by a team in Shantong University…

Sidetrack: Encryption



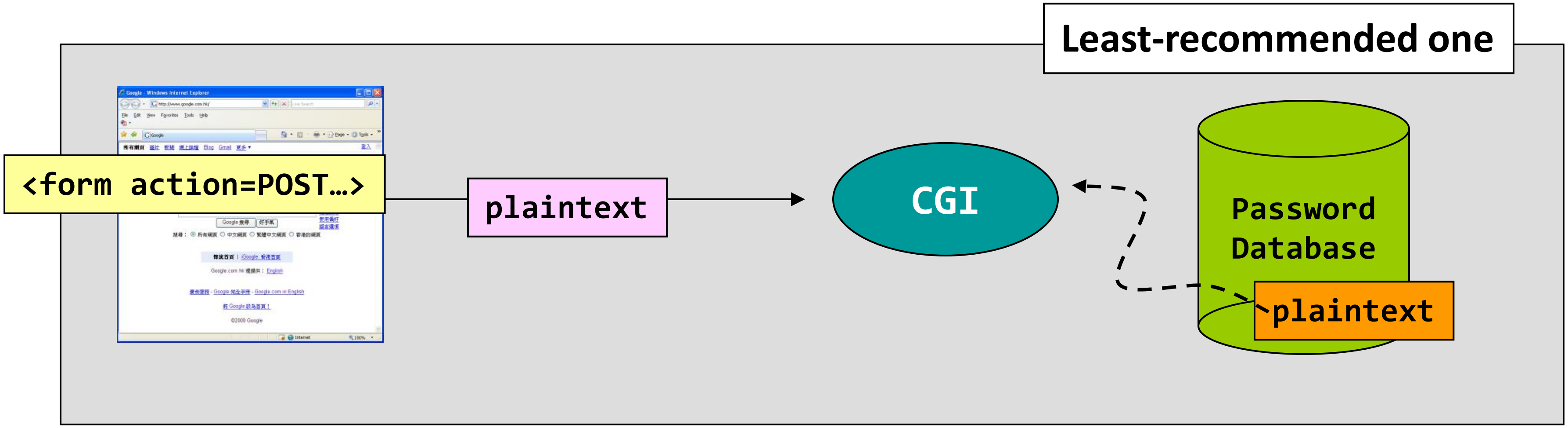
Encryption and decryption only work when you provide the functions **a valid encryption key**.

Encrypted value can be reversed by decryption...only when the same encryption key is supplied to decryption function.

Want to know more? Take ENGG 5105!

Password transfer & management

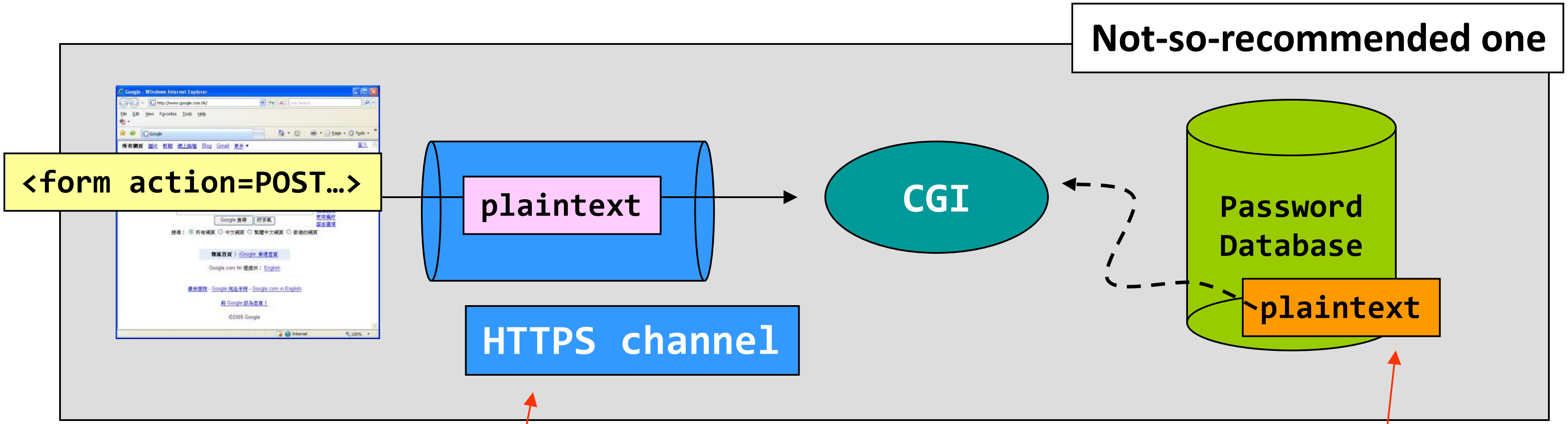
- Version 1



This method does not require further introduction…

Password transfer & management

- Version 2

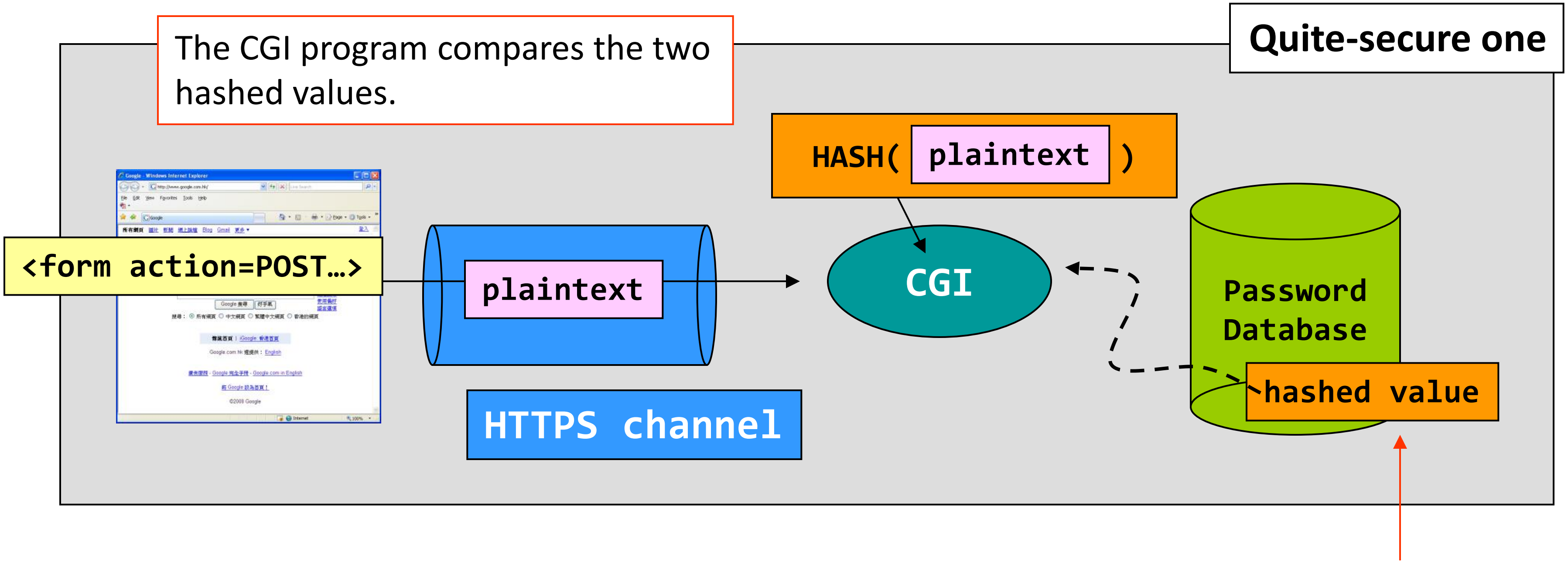


Whether HTTPS is available or not depends on the web server…

Still, it is not safe to store plaintext password…

Password transfer & management

- Version 3

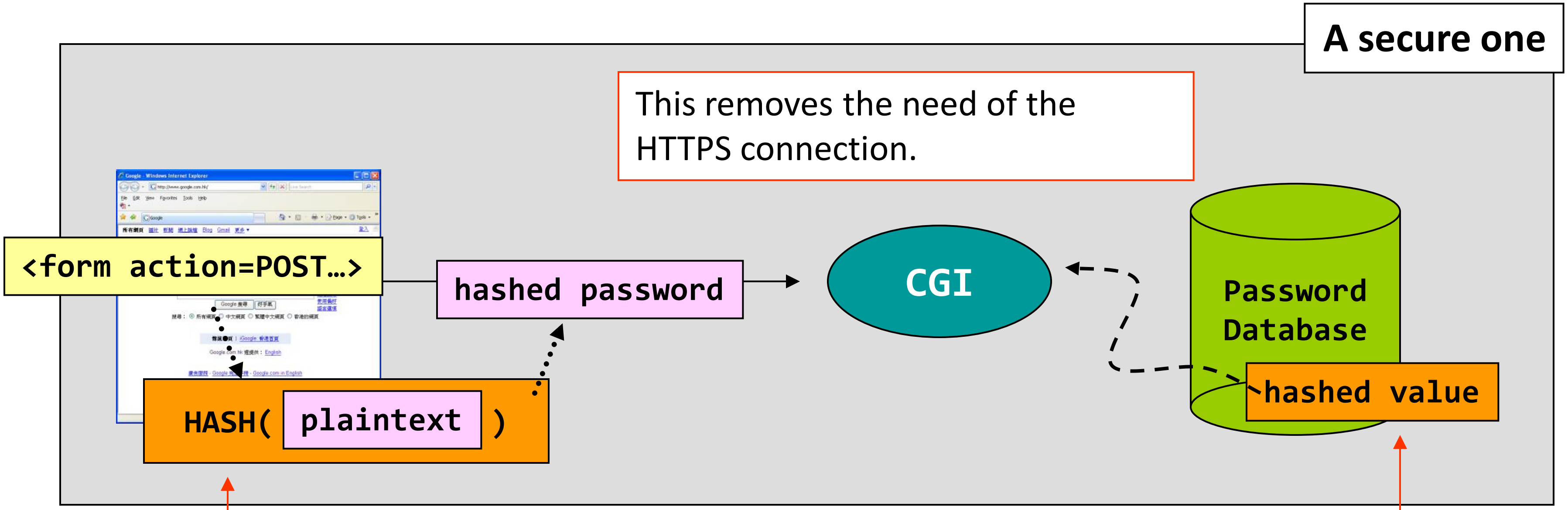


The CGI program compares the two hashed values.

There are many famous hashing algorithm, e.g., MD5, SHA-1, etc…

Password transfer & management

- Version 4



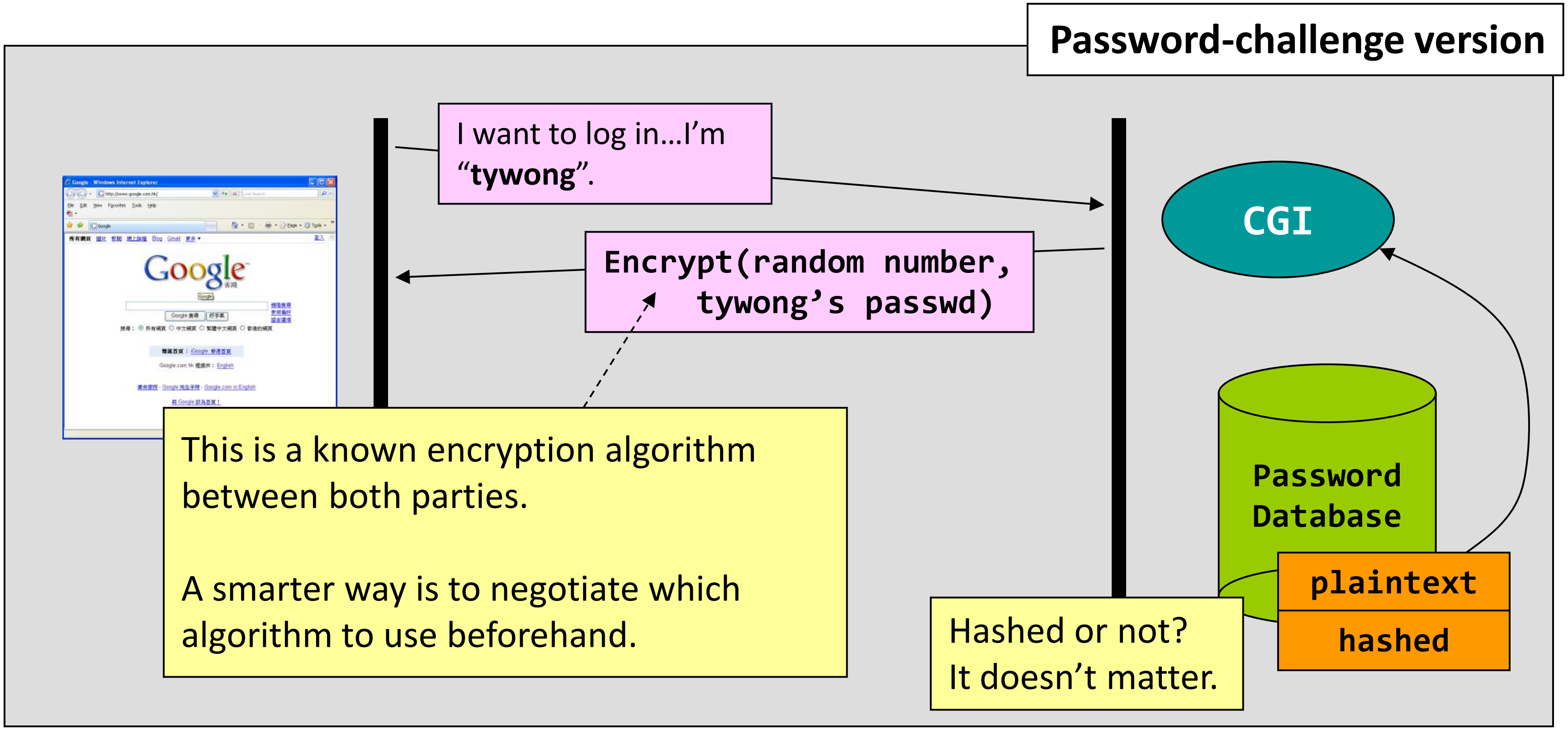
This removes the need of the HTTPS connection.

The browser does not have the capability to do hashing…we need the help from **JavaScript**.

There are many famous hashing algorithm, e.g., MD5, SHA-1, etc…

Password transfer & management

- Version 5



Password-challenge version

I want to log in…I’m “**tywong**”.

```
Encrypt(random number,
  tywong’s passwd)
```

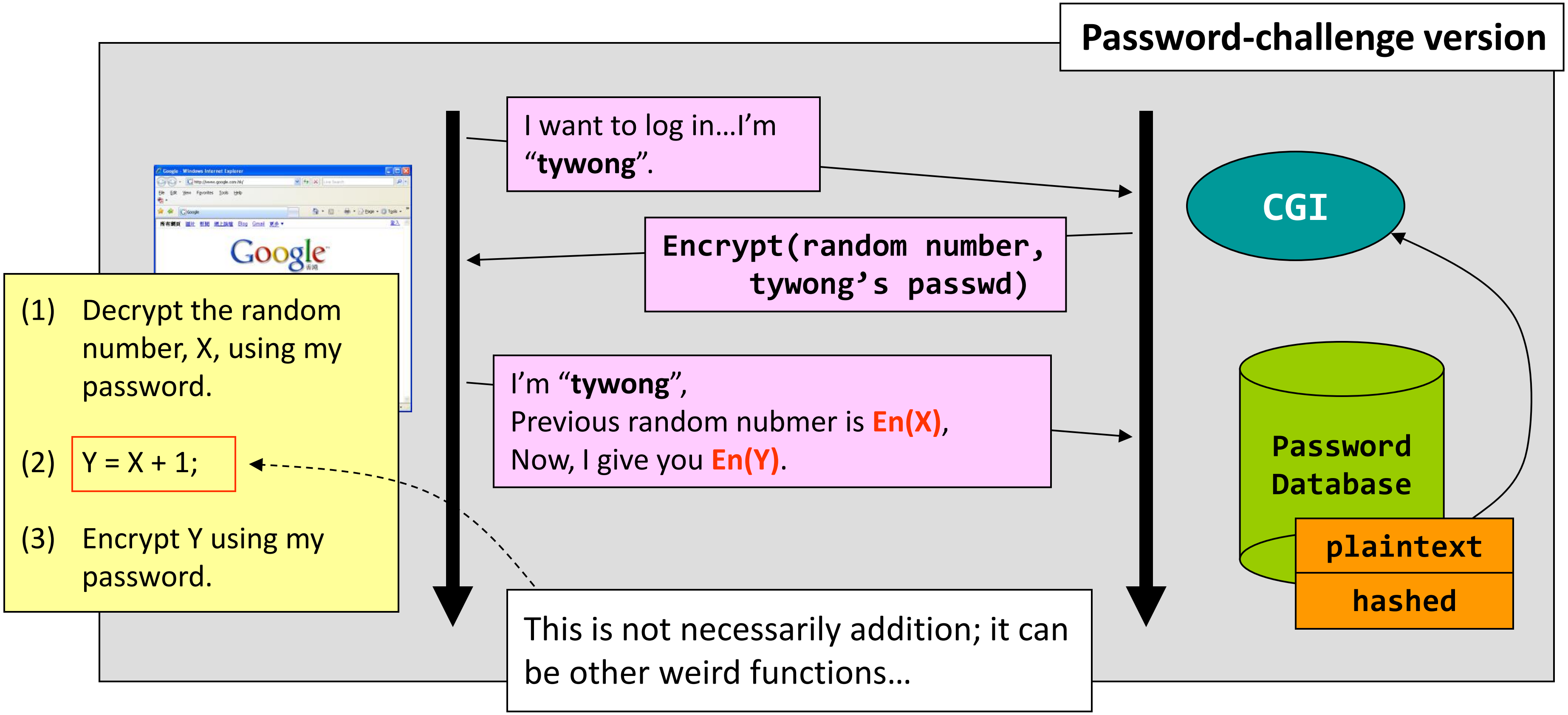
This is a known encryption algorithm between both parties.

A smarter way is to negotiate which algorithm to use beforehand.

Hashed or not?
It doesn’t matter.

Password transfer & management

- Version 5



Password-challenge version

I want to log in…I’m “**tywong**”.

Encrypt(random number, tywong’s passwd)

I’m “**tywong**”,
Previous random nubmer is **En(X)**,
Now, I give you **En(Y)**.

(1) Decrypt the random number, X, using my password.

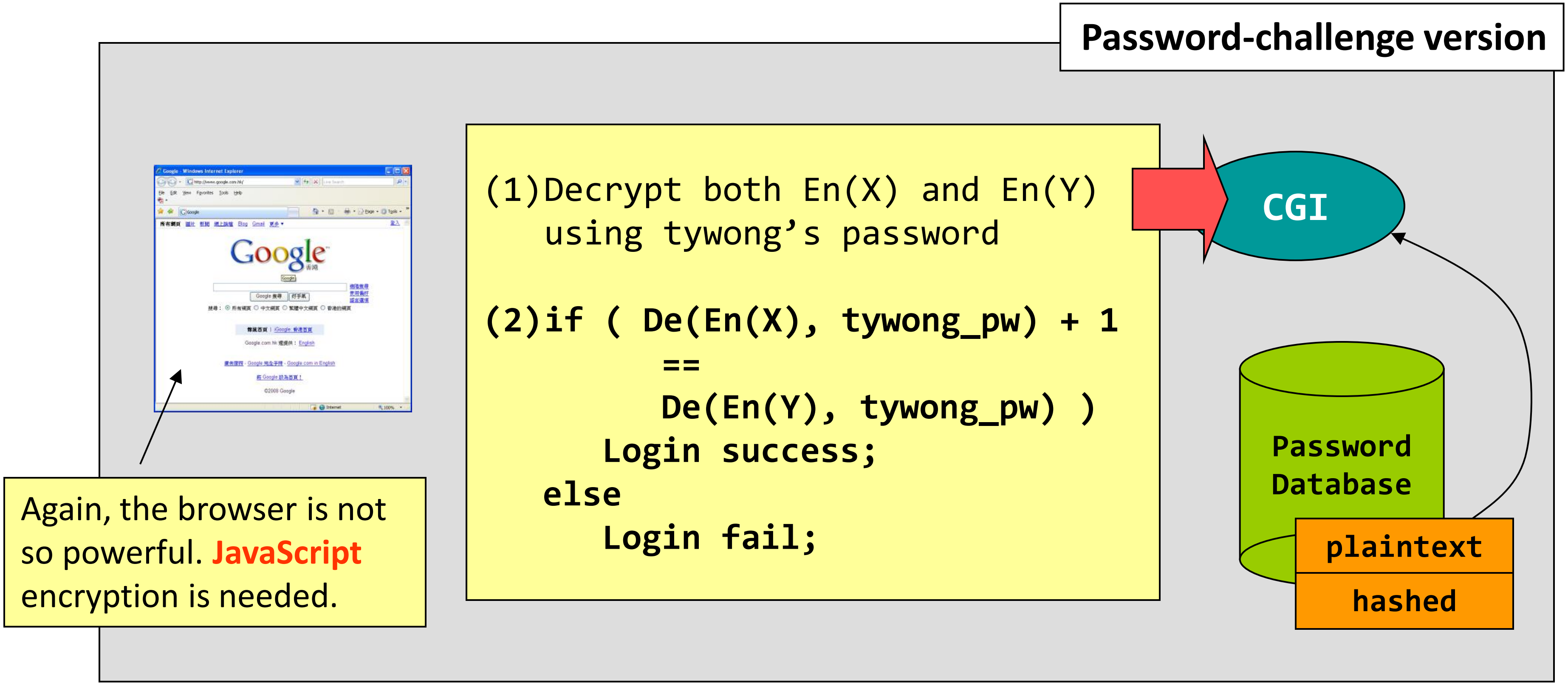
(2) Y = X + 1;

(3) Encrypt Y using my password.

This is not necessarily addition; it can be other weird functions…

Password transfer & management

- Version 5



```
(1)Decrypt both En(X) and En(Y)
   using tywong's password

(2)if ( De(En(X), tywong_pw) + 1
          ==
        De(En(Y), tywong_pw) )
      Login success;
   else
      Login fail;
```

Again, the browser is not so powerful. JavaScript encryption is needed.

Password transfer & management

- It involves the transmission and the storage of the password.

Transmission	Storage	Comments
plain-text	plain-text	Seldom used; or, in toy systems…
plain-text	hashed	**Mostly deployed**; usually together with HTTPS.
hashed	hashed	Not much…
challenged	hashed	High-end systems only. Typically, those are not web-based system.

We always want to have a simple client because the client side is **not (so) trustworthy nor powerful enough**…so if the last two choices are needed, they will be implemented externally, such as a Java applet…

Password Management?

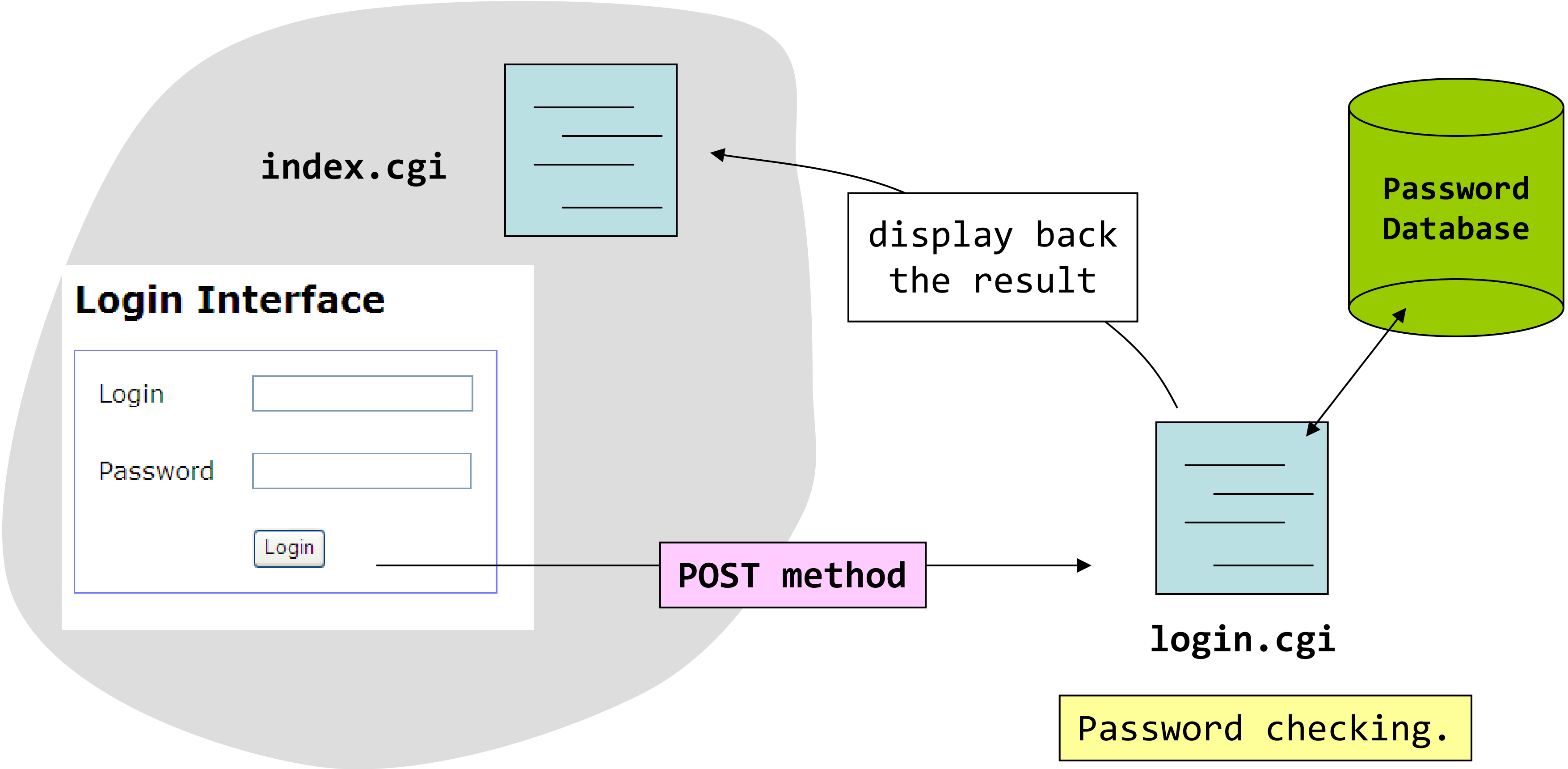
- It involves the transmission and the storage of the password.

Transmission	Storage	Comments
plain-text	plain-text	Seldom used; or, in toy systems…
plain-text	hashed	**Mostly deployed**; usually together with HTTPS.
hashed	hashed	Not much…
challenged	hashed	High-end systems. Typically, those are not web-based system.

By the way, do you have any idea on the storage side?

Let’s start with a toy system…

- This is really a toy…



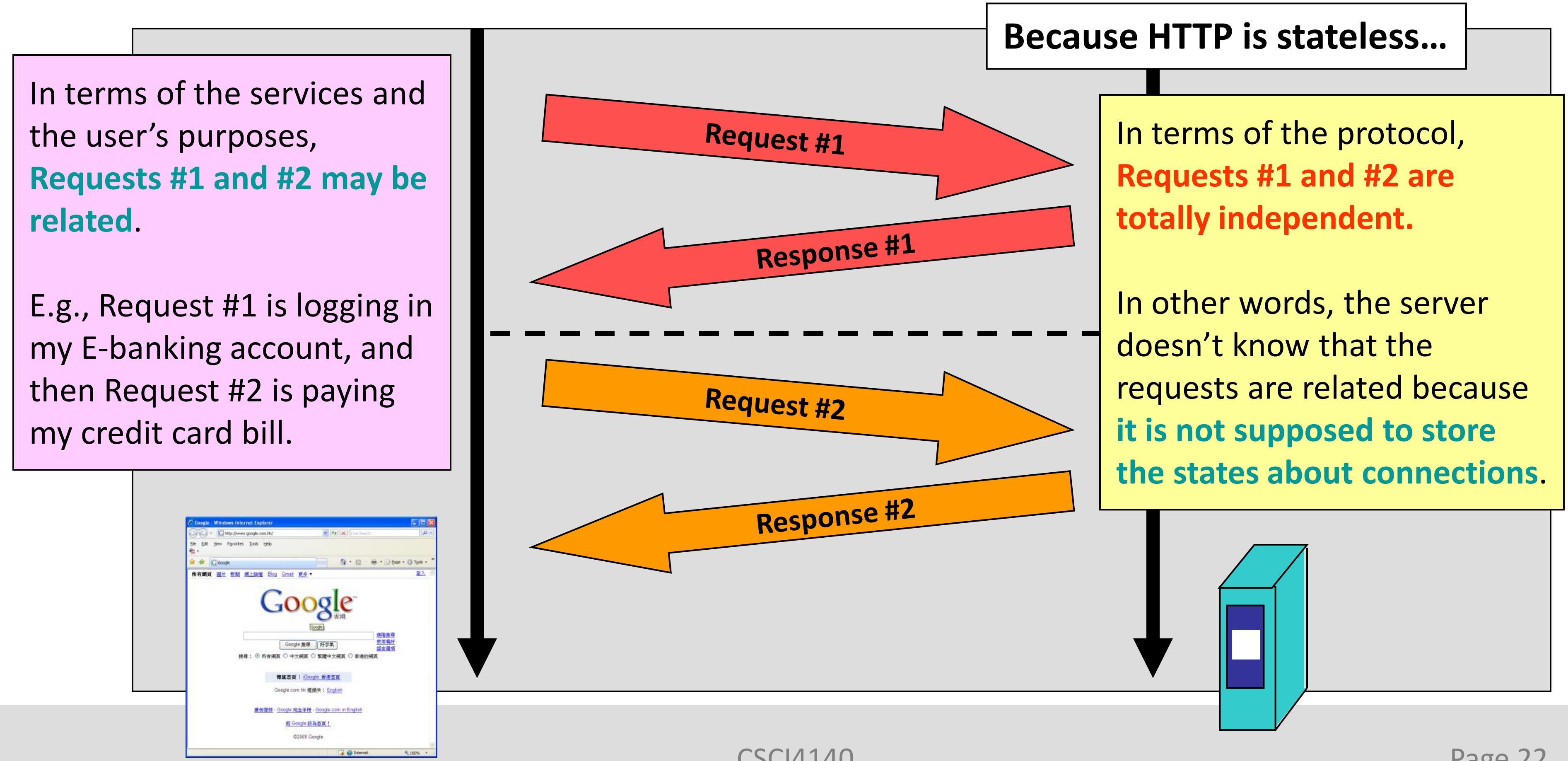
[Example] “hashed_passwd/”

Session Management

- The What, The How, and The Why.

Our challenge…

- Session Management…
  - Why do we need that?



Because HTTP is stateless…

In terms of the services and the user's purposes, **Requests #1 and #2 may be related**.

E.g., Request #1 is logging in my E-banking account, and then Request #2 is paying my credit card bill.

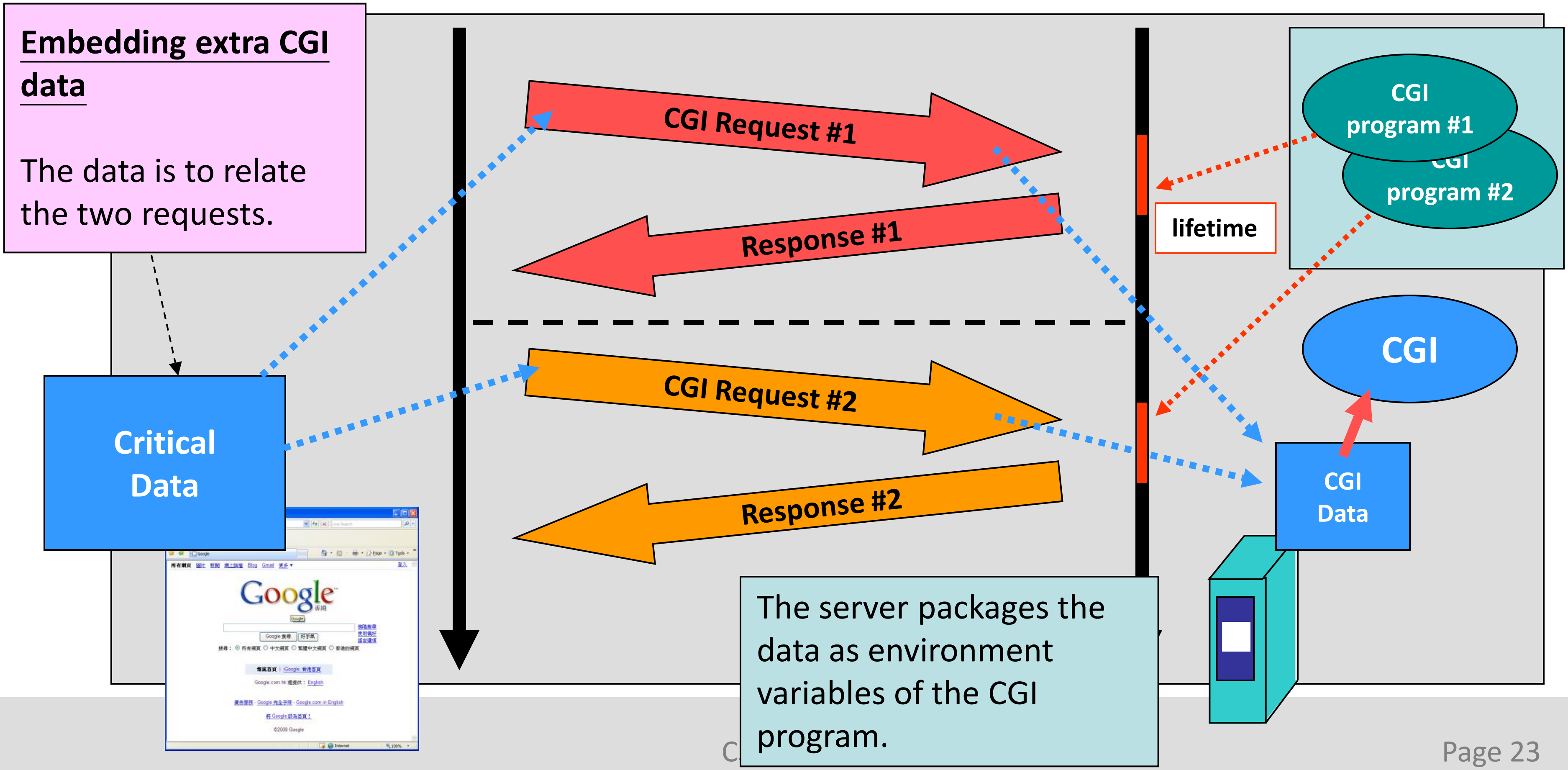
In terms of the protocol, **Requests #1 and #2 are totally independent.**

In other words, the server doesn't know that the requests are related because **it is not supposed to store the states about connections**.

Our challenge…

The physical meaning is: **we may not have a program always running and serves all requests**. Under most cases, different processes serve different requests independently.

Nonetheless…CGI programs are short-lived.

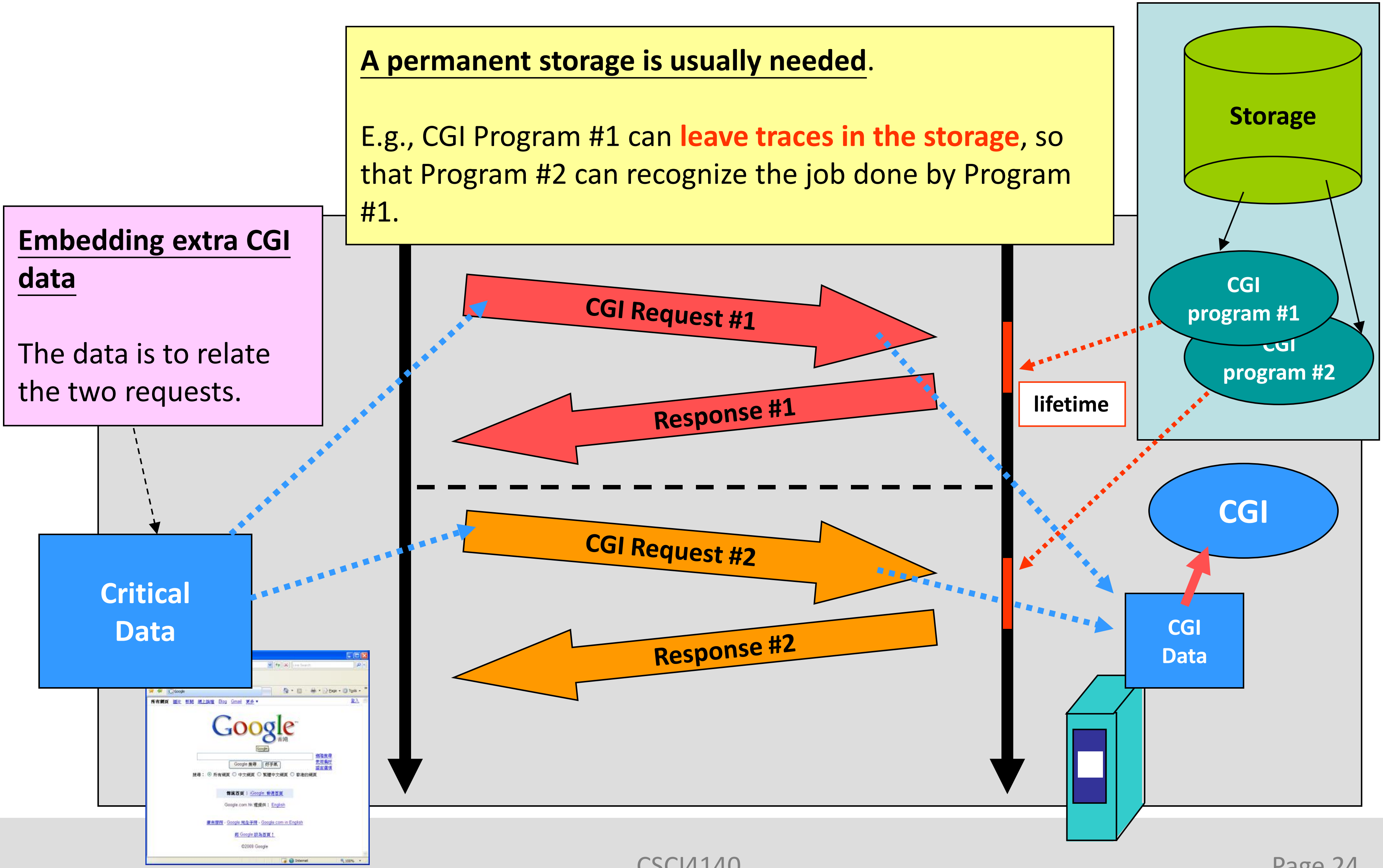


Embedding extra CGI data

The data is to relate the two requests.

The server packages the data as environment variables of the CGI program.

Our challenge…



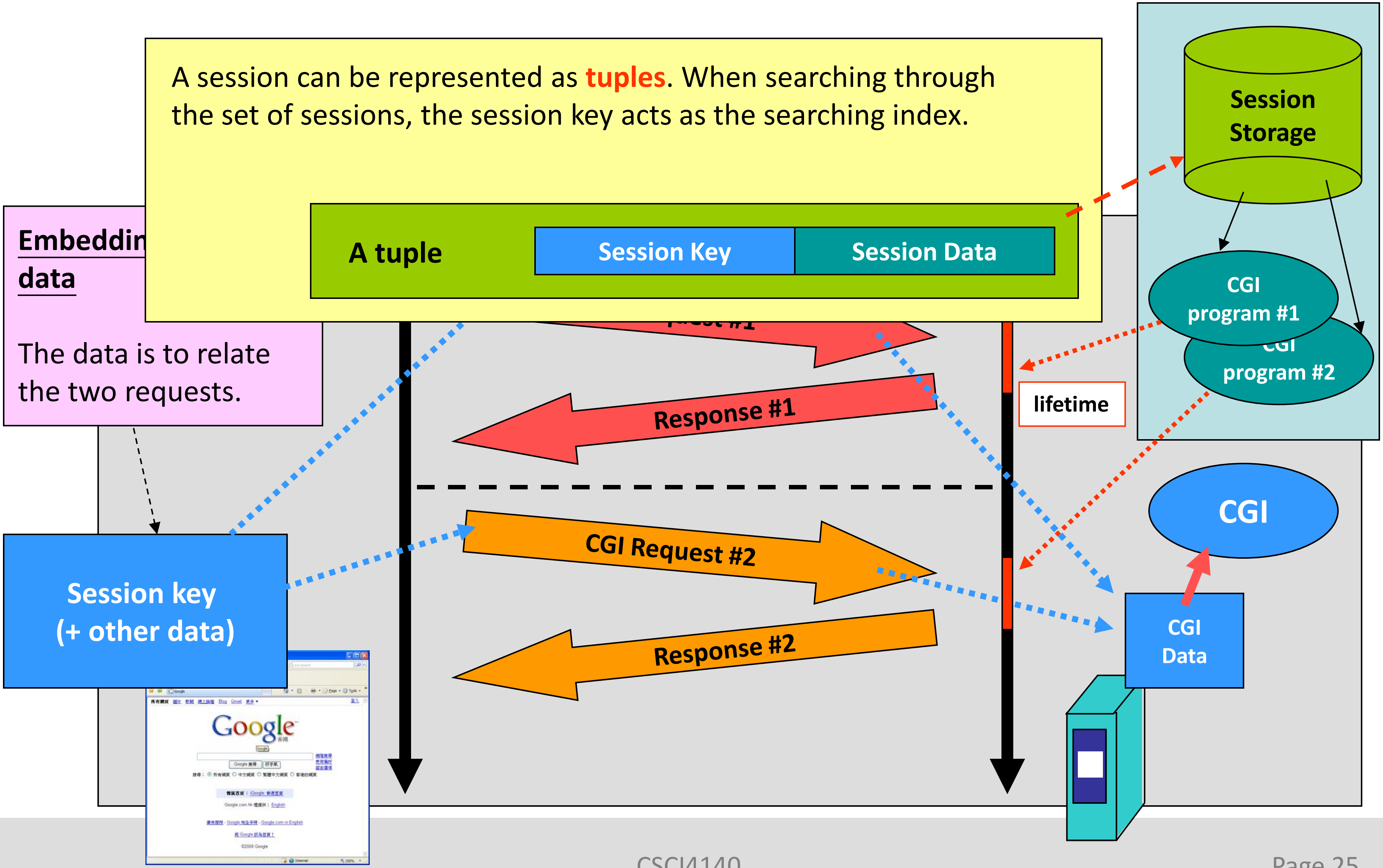
A permanent storage is usually needed.

E.g., CGI Program #1 can leave traces in the storage, so that Program #2 can recognize the job done by Program #1.

Embedding extra CGI data

The data is to relate the two requests.

Session management?



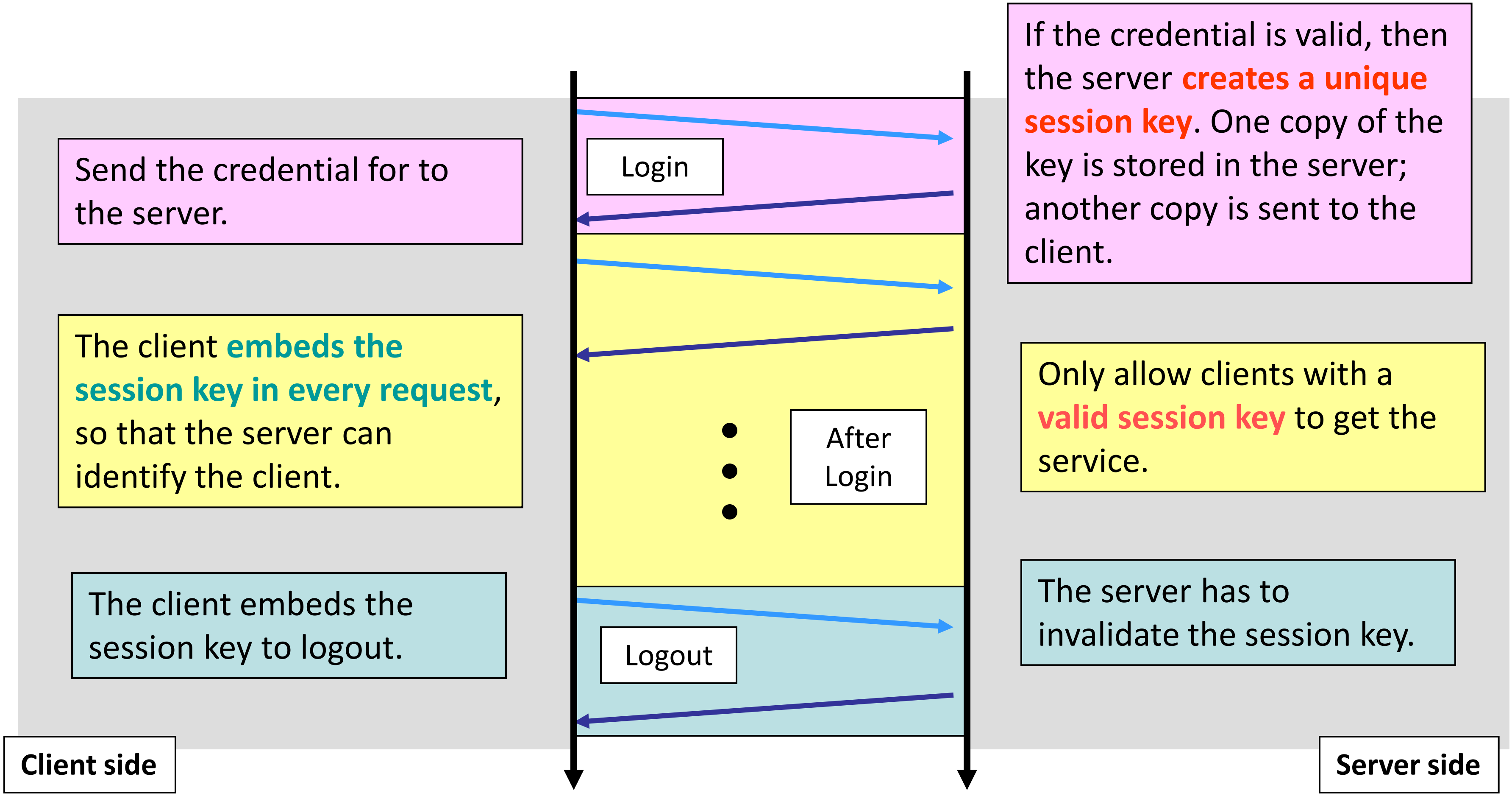
A session can be represented as **tuples**. When searching through the set of sessions, the session key acts as the searching index.

Embeddin data

The data is to relate the two requests.

Session management!

- Now, we have a clear goal:



Send the credential for to the server.

If the credential is valid, then the server **creates a unique session key**. One copy of the key is stored in the server; another copy is sent to the client.

The client **embeds the session key in every request**, so that the server can identify the client.

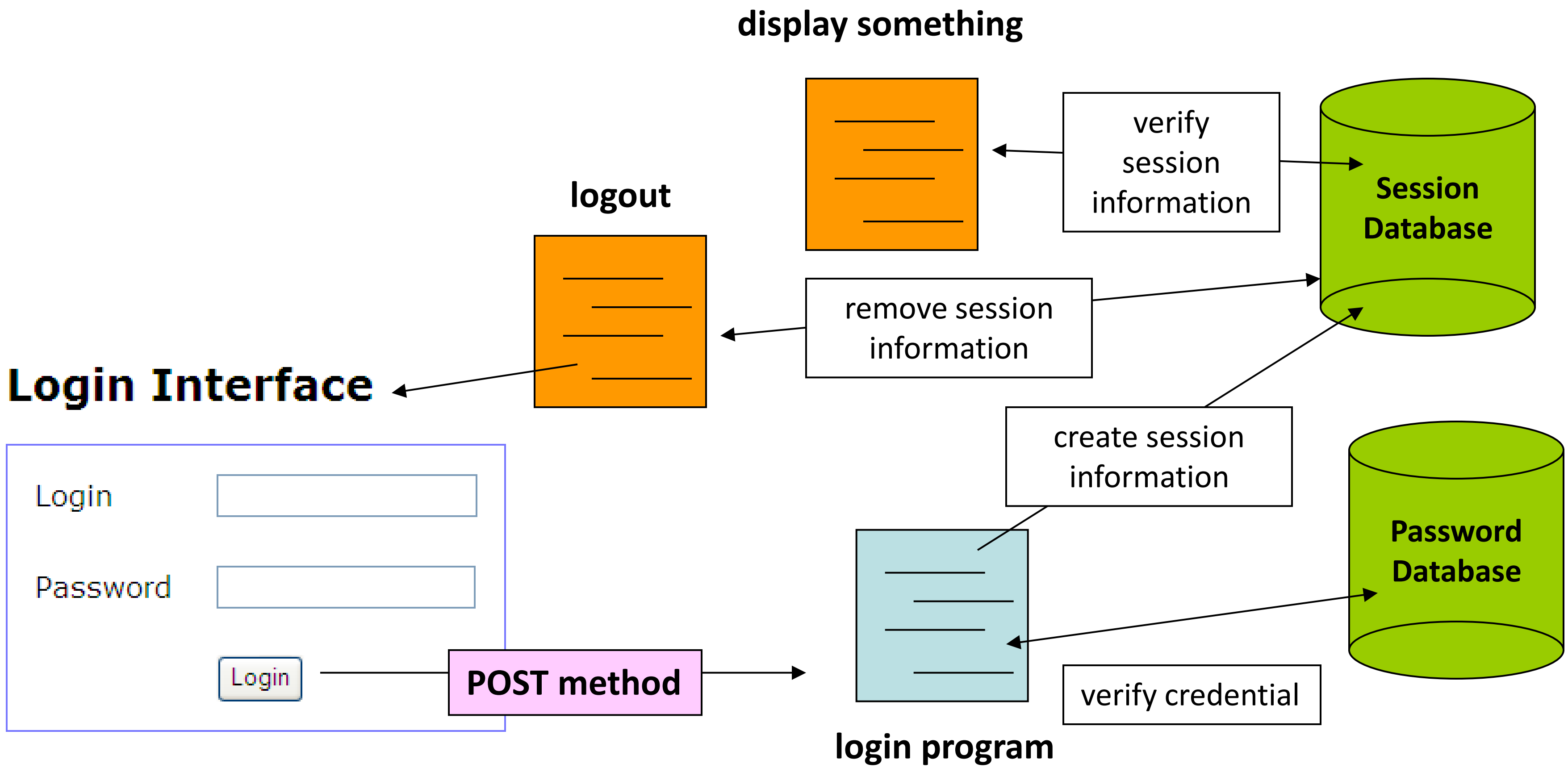
Only allow clients with a **valid session key** to get the service.

The client embeds the session key to logout.

The server has to invalidate the session key.

Session management – Example

- We extend the toy presented previously…



Session management…

- Class discussions:
  - How bad is reloading?
  - Bookmarking a URL with the session key?
  - Passing the session key between pages?
  - Returning clients?

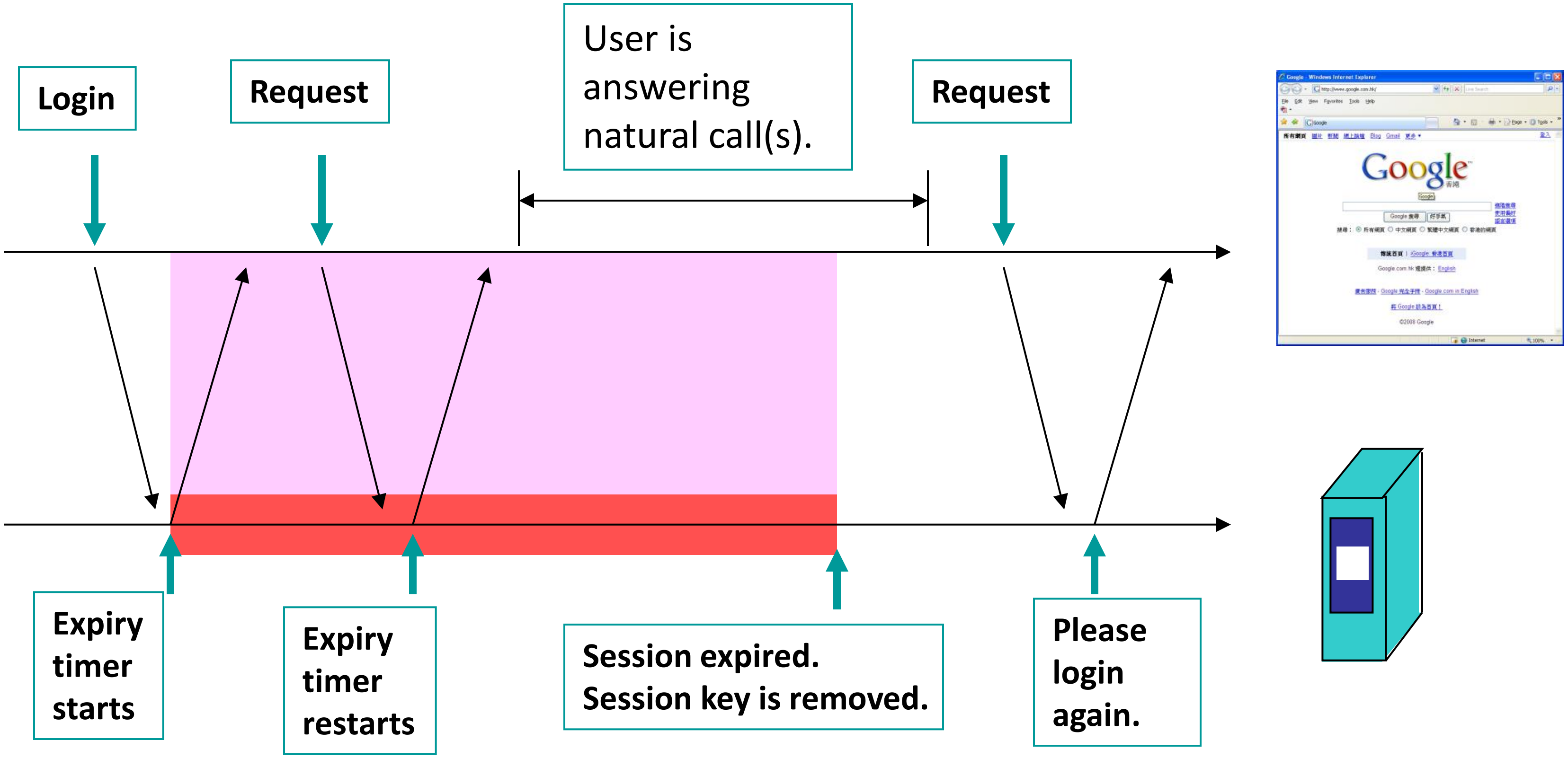
Try reloading the page…

Let’s look at the hyperlink…

[Example] “bad_design_1/”, “bad_design_2/”, “good_design/”

Design Issues

- Session and Session key.
  - Problem #1: **Duration** – expiring mechanism.



Login

Request

User is answering natural call(s).

Request

Expiry timer starts

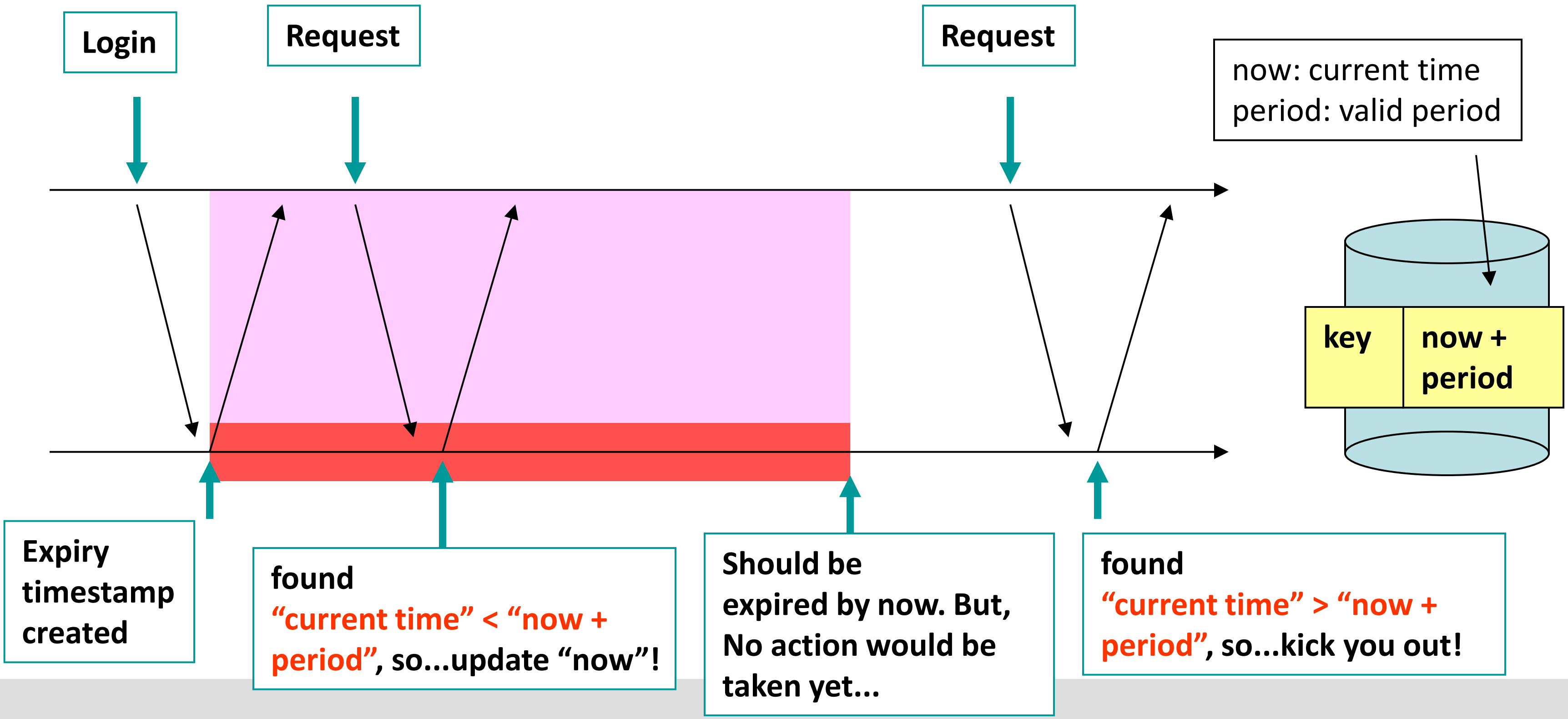
Expiry timer restarts

Session expired. Session key is removed.

Please login again.

Design Issues

- Session and Session key.
  - Problem #1: **Duration** – expiring mechanism.
    - However, we know that CGI programs are **not long-lasting**…then, how can we implement the timer-like feature?

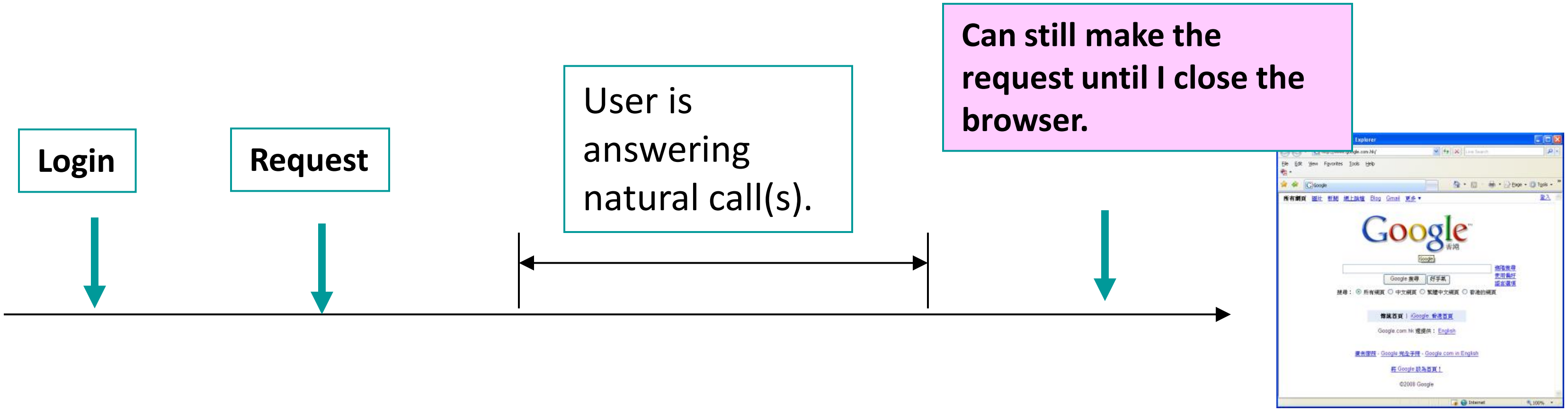


Login
Request
Request
now: current time
period: valid period
key
now + period
Expiry timestamp created
found "current time" < "now + period", so...update "now"!
Should be expired by now. But, No action would be taken yet...
found "current time" > "now + period", so...kick you out!

Design Issues

- Session and Session key.
  - Problem #1: **Duration** – expiring mechanism.
    - Or, can the expiry be controlled **on the client side**?
    - Again, how?

HTTP Cookies, and we will learn about it later!



Login

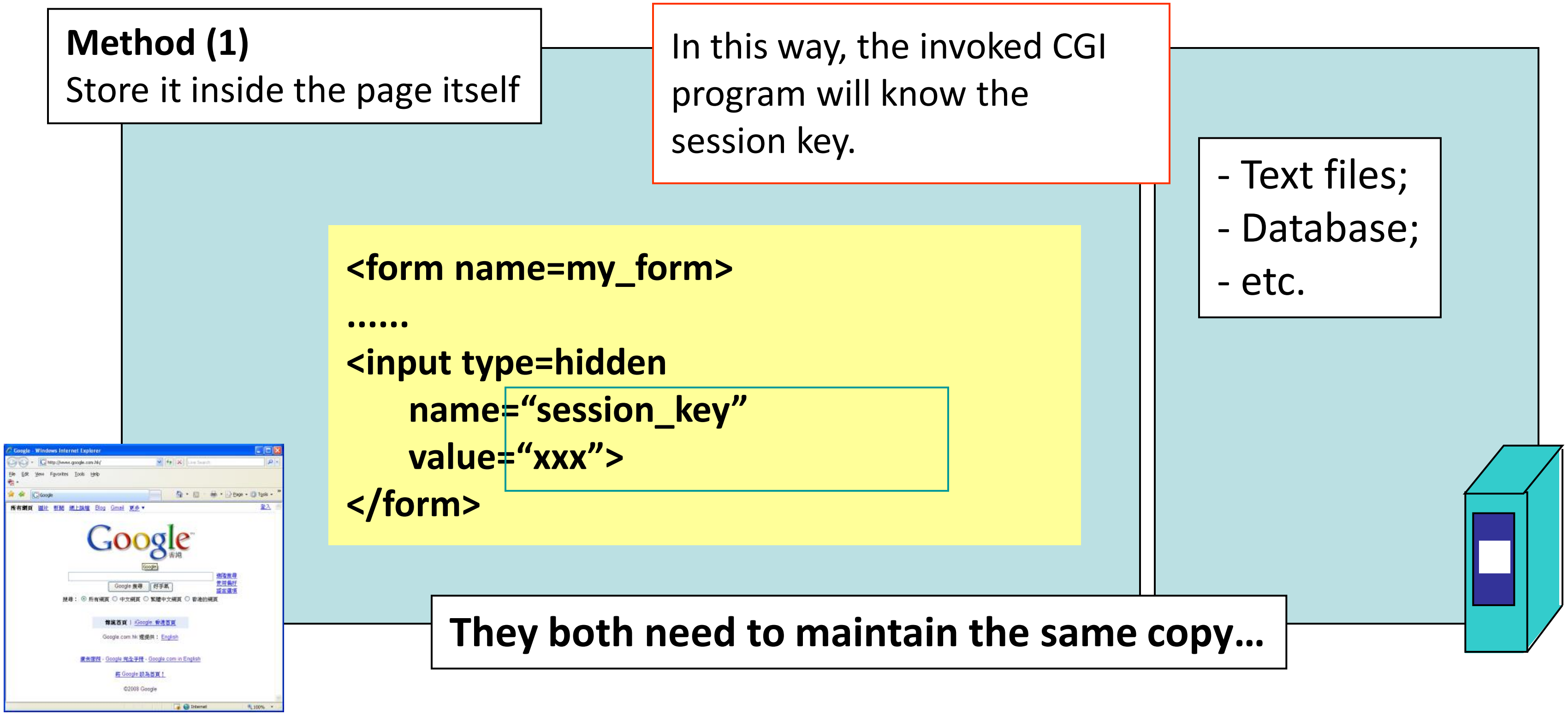
Request

User is answering natural call(s).

Can still make the request until I close the browser.

Design Issues

- Session and Session key.
  - Problem #2: **Storage** – who and where?



Method (1)
Store it inside the page itself

In this way, the invoked CGI program will know the session key.

- Text files;
- Database;
- etc.

```
<form name=my_form>
......
<input type=hidden
      name="session_key"
      value="xxx">
</form>
```

They both need to maintain the same copy…

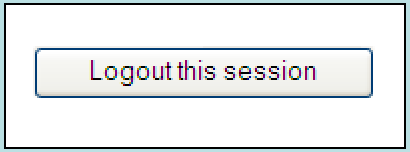
Design Issues

- Session and Session key.
  - Problem #2: **Storage** – who and where?

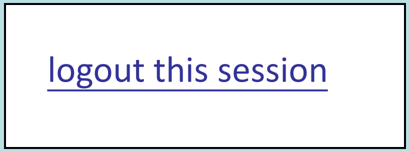
Method (1) - drawback

The web system must use the **submit function**.

```
<form ......>
<input type=hidden ......>
<input type=submit value="Logout this session">
</form>
```

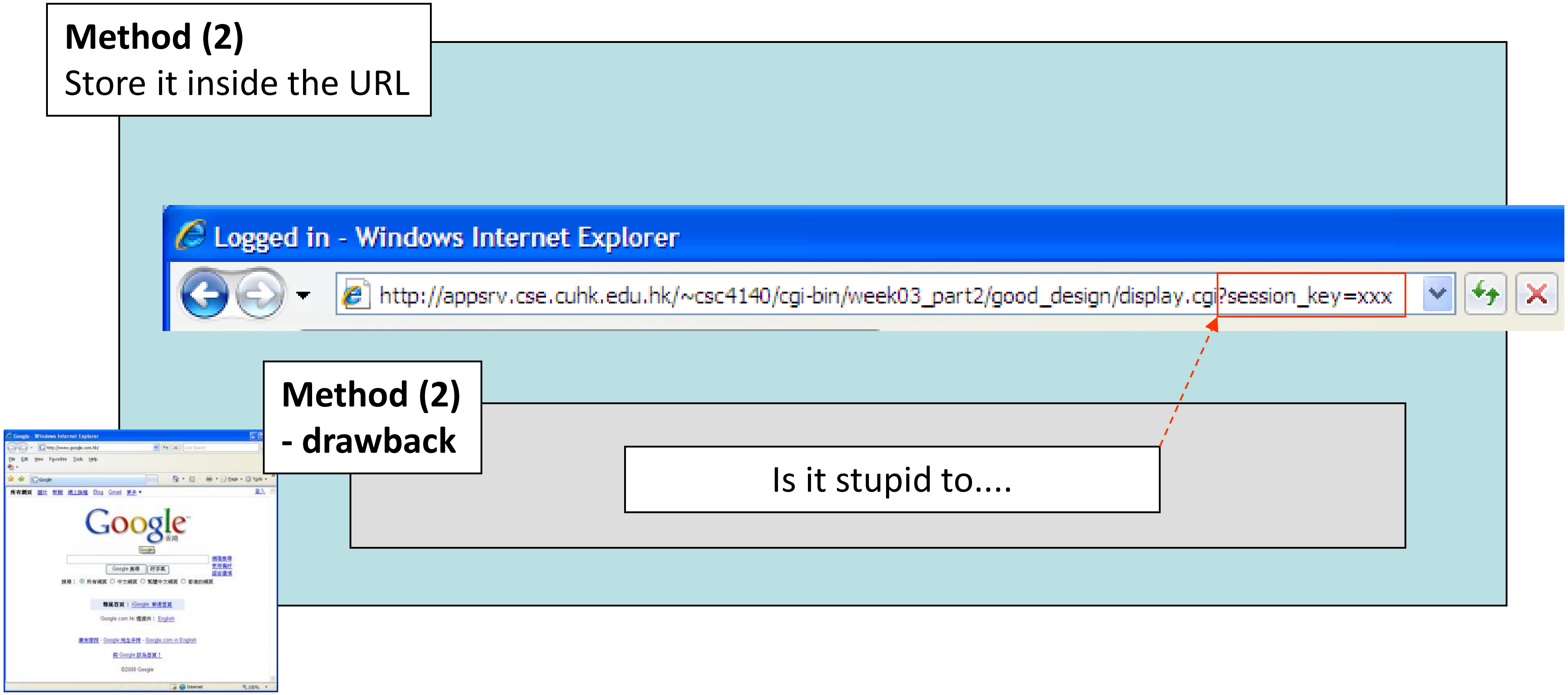


```
<form ......>
<input type=hidden ......>
<a href="javascript: document.my_form.submit();">
logout this session</a>
</form>
```



Design Issues

- Session and Session key.
  - Problem #2: **Storage** – who and where?

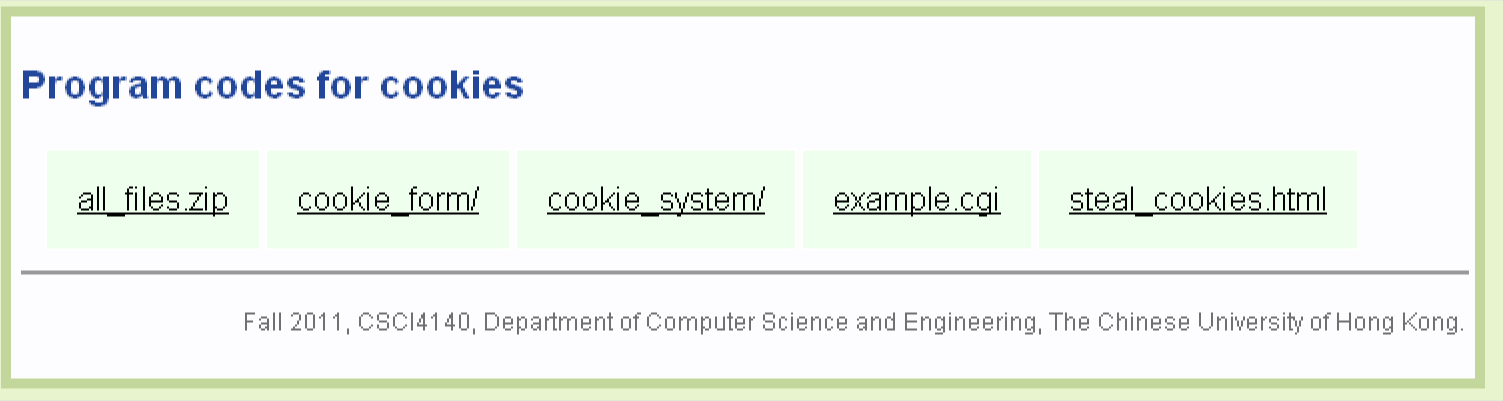


Method (2)
Store it inside the URL

http://appsrv.cse.cuhk.edu.hk/~csc4140/cgi-bin/week03_part2/good_design/display.cgi?session_key=xxx

Method (2)
- drawback

Is it stupid to....



Program codes for cookies

all_files.zip | cookie_form/ | cookie_system/ | example.cgi | steal_cookies.html

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

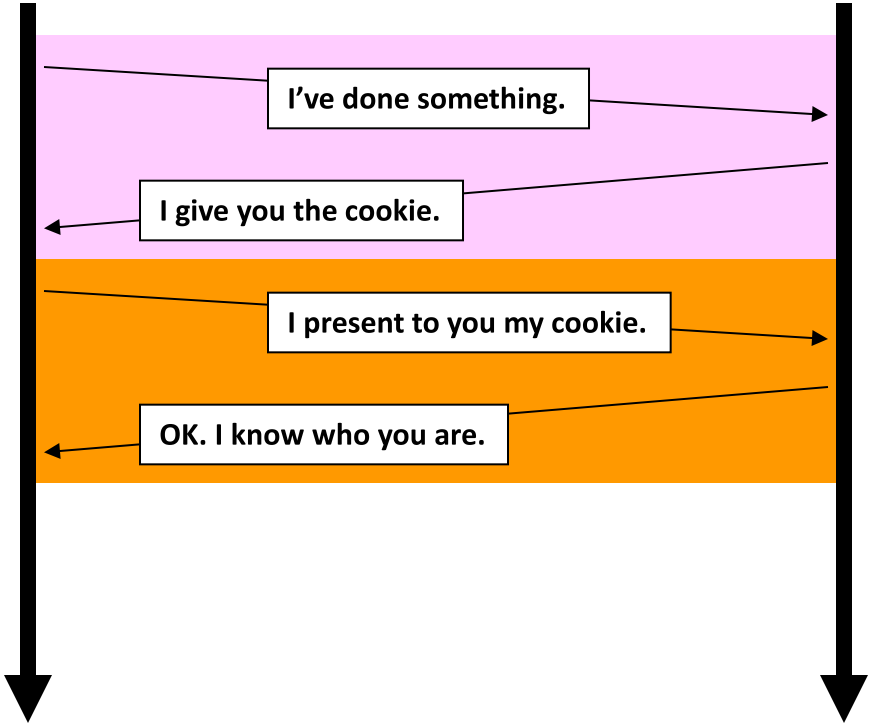
http://demo4140-tywong.rhcloud.com/04_cookies/

Session Management

- The HTTP Cookies

The Finishing Touch – Cookie

- Three aspects about cookies:
  - Set, Save, and Send.



I've done something.

I give you the cookie.

I present to you my cookie.

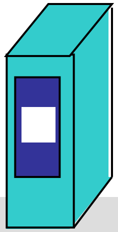
OK. I know who you are.

(1) Set cookies by server

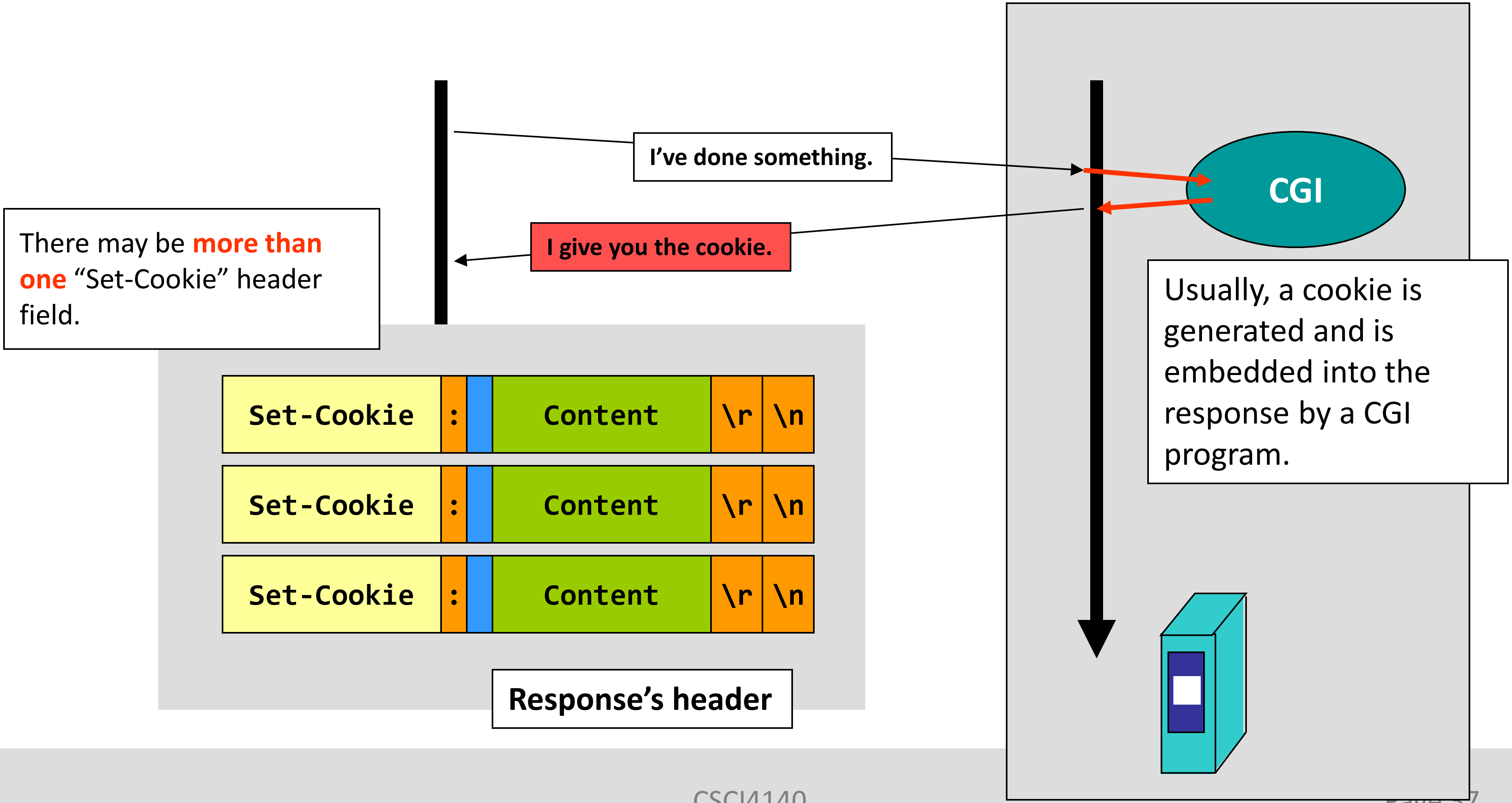
(2) Receive and save cookies on client

(3) Cookies received by server

Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
Page
Tools
所有網頁 圖片 新聞 網上論壇 Blog Gmail 更多
登入
Google
香港
Google 搜尋
好手氣
Google.com.hk 還提供： English
Google.com in English
©2008 Google
Internet
100%



What is HTTP Cookie?



I've done something.

I give you the cookie.

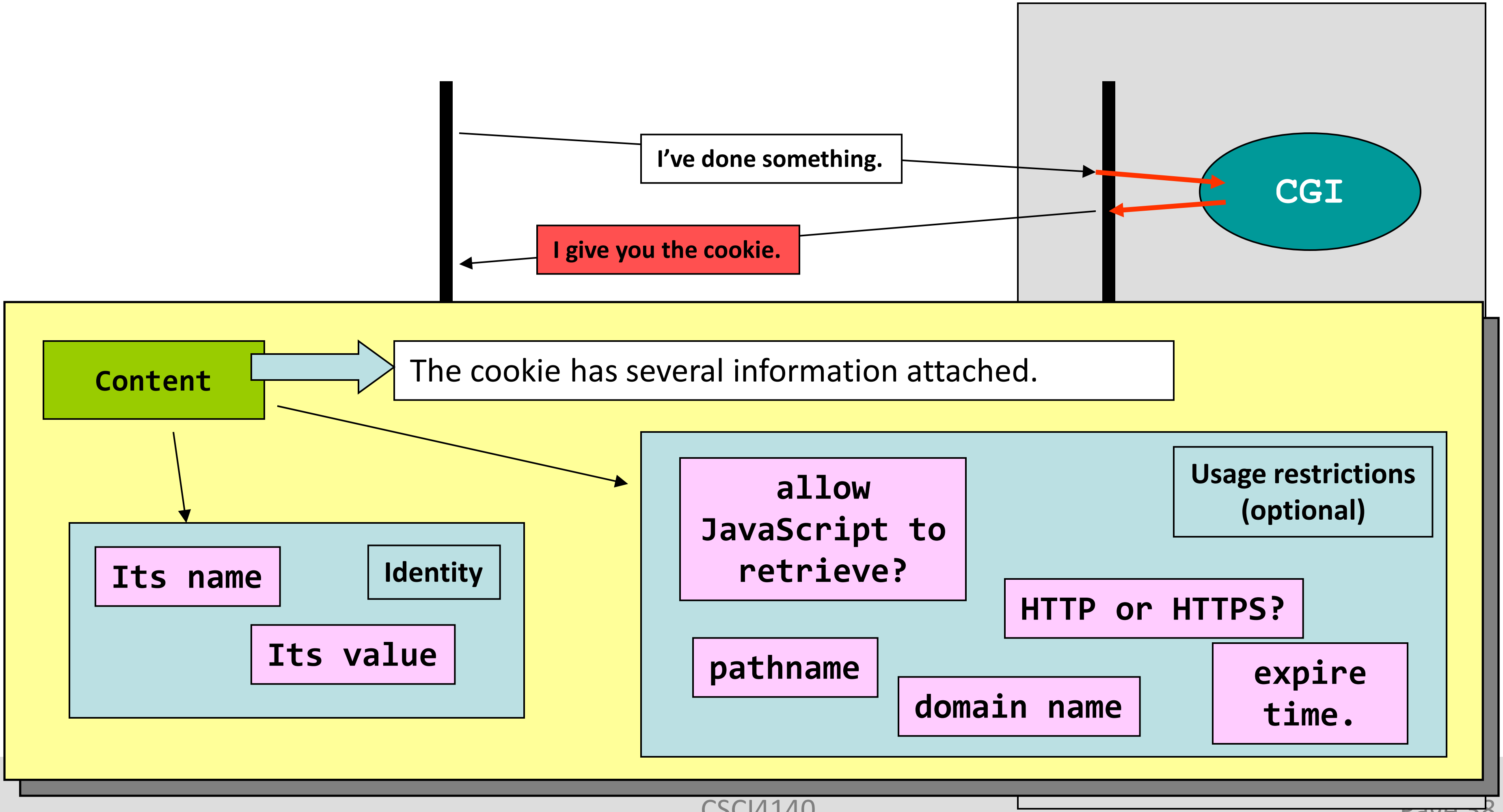
There may be **more than one** "Set-Cookie" header field.

Set-Cookie	:		Content	\r	\n
Set-Cookie	:		Content	\r	\n
Set-Cookie	:		Content	\r	\n

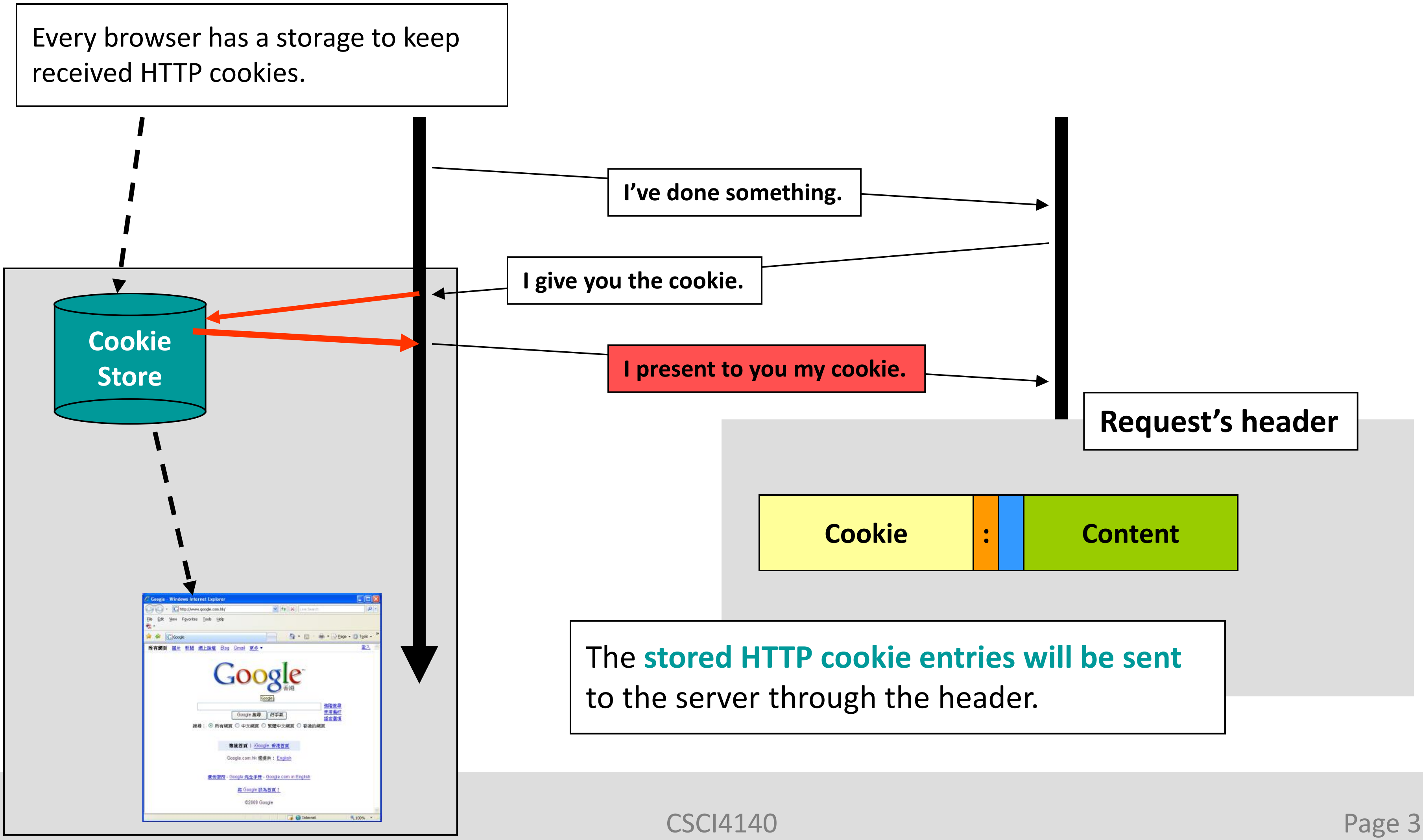
Response's header

Usually, a cookie is generated and is embedded into the response by a CGI program.

What is HTTP Cookie?



What is HTTP Cookie?



Every browser has a storage to keep received HTTP cookies.

I've done something.

I give you the cookie.

I present to you my cookie.

Request's header

Cookie	:	Content

The **stored HTTP cookie entries will be sent** to the server through the header.

What is HTTP Cookie?

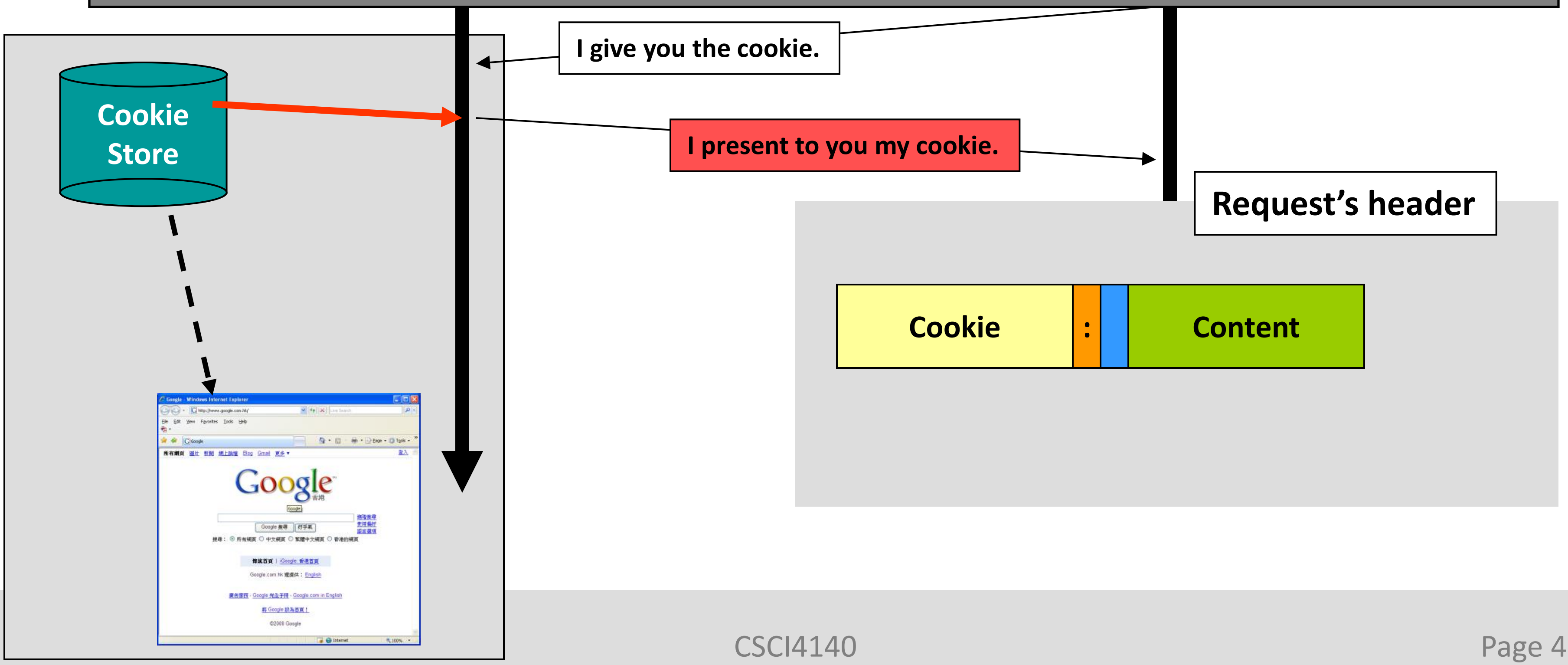
The browser has to decide which cookie(s) to send for each GET/POST message based on the following criteria:

(1) Does the domain (of the remote site) match?

(2) Does the pathname match?

(3) Expired?

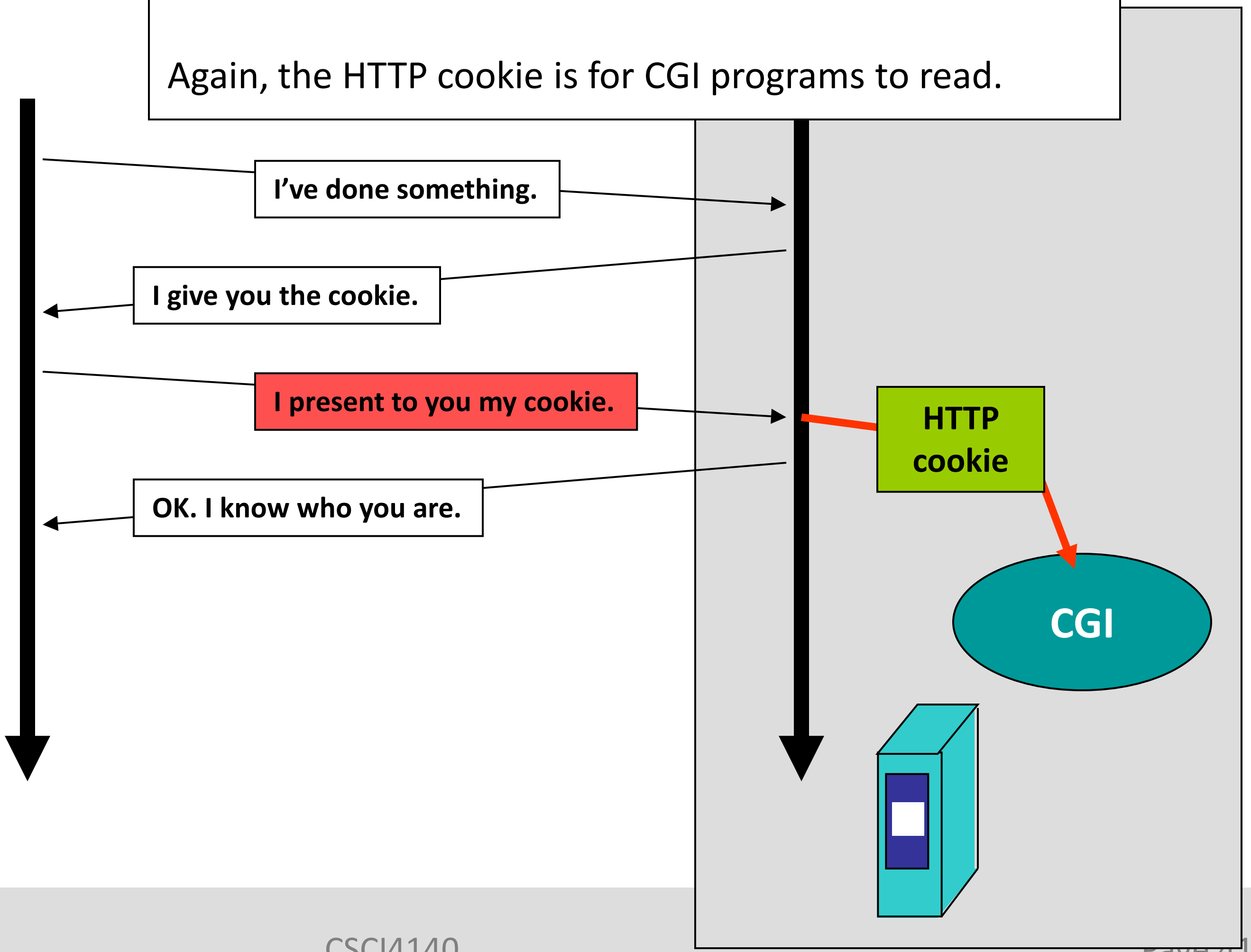
(4) HTTP or HTTPS?



What is HTTP Cookie?

Of course, the HTTP server doesn't know what the cookie is about.

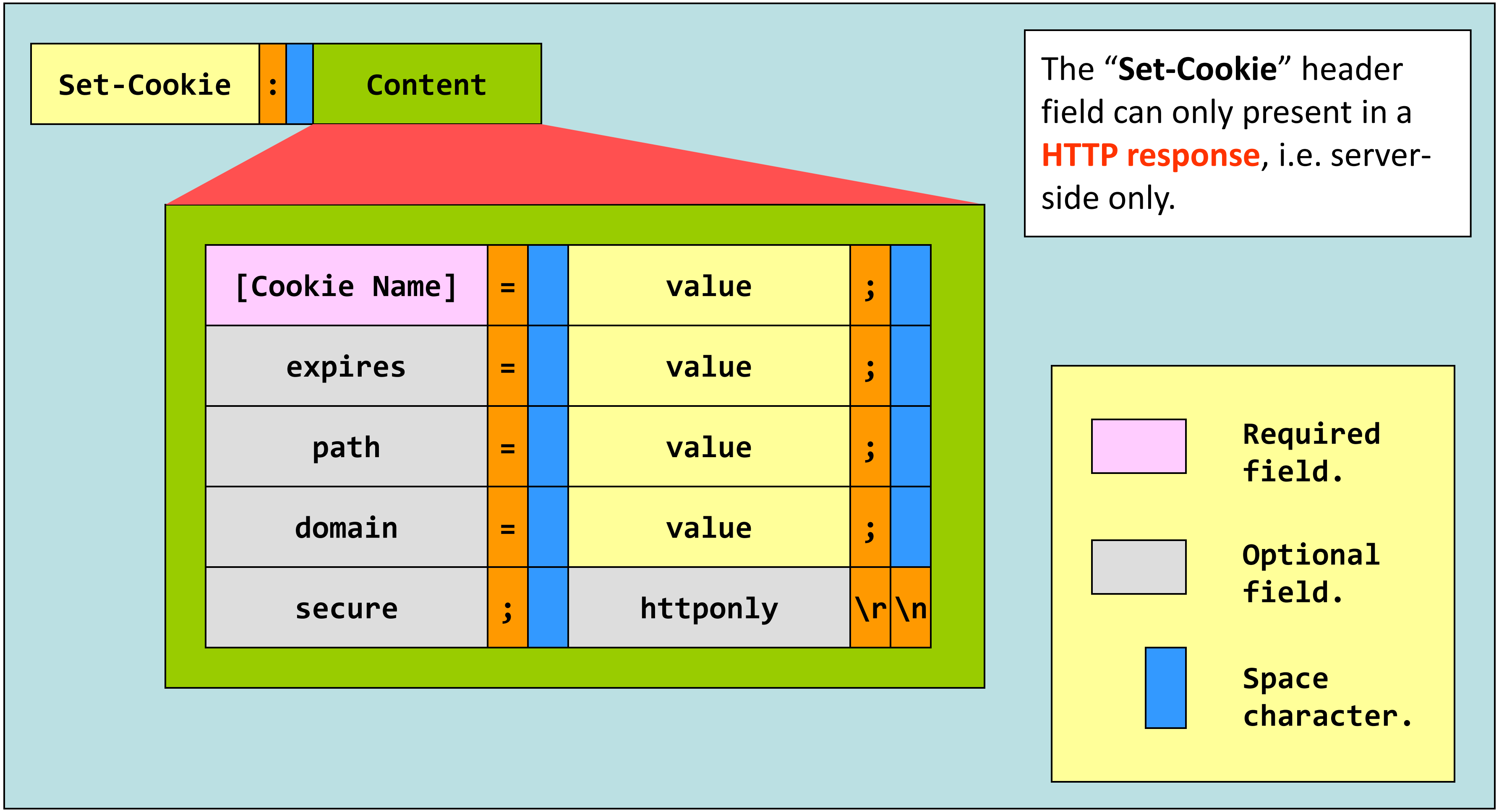
Again, the HTTP cookie is for CGI programs to read.



HTTP Cookie

– the bits and the bytes...

Set-Cookie Format



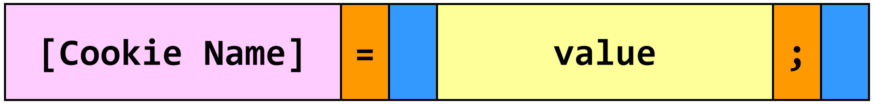
Set-Cookie	:	🟦	Content

[Cookie Name]	=	🟦	value	;	🟦
expires	=	🟦	value	;	🟦
path	=	🟦	value	;	🟦
domain	=	🟦	value	;	🟦
secure	;	🟦	httponly	\r	\n

The "**Set-Cookie**" header field can only present in a **HTTP response**, i.e. server-side only.

🟪	Required field.
⬜	Optional field.
🟦	Space character.

Set-Cookie Format - Name



The name is the **unique identifier** for each HTTP cookie.

Remember, this is part of a HTTP request!

So, both the name and the values should be encoded in the **URI format**.

brain damage ✗

brain%20damage ✓

By the way, how to do such an encoding in Perl?

Set-Cookie Format - Name

- How to do the encoding in Perl?

For details, http://perldoc.perl.org/

1st thing you need to know.

In Perl, "**ord()**" prints the numeric representation of a character.

ord() – reverse of chr()

2nd thing you need to know.

It is safe to encode all characters.

E.g., You can use "**%20**" instead of '**+**' to represent a space character, and it is safe to do so.

3rd thing you need to know.

We have "**sprintf()**" in Perl, but..

The return value is the produced string, unlike its counterpart in C.

E.g., **$output = sprintf("%d", $input);**

Set-Cookie Format - Name

- How to do the encoding in Perl?

For details, http://perldoc.perl.org/

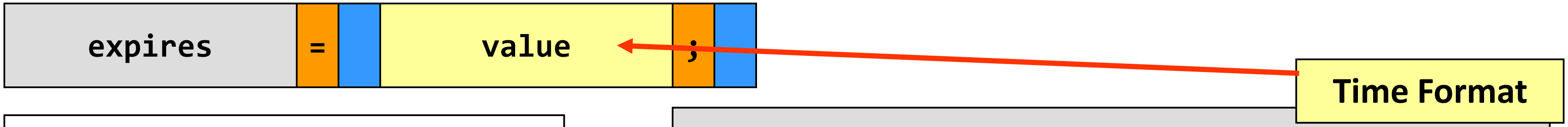
s	‘.’ matches everything including newline.
e	Execute commands before matching/substituting.
g	Replace globally, i.e. all occurrences

Answer:

```
$input =~ s/(.)/sprintf("%%%02X", ord($1))/seg;
```

```
$input =~ s/([^A-Za-z0-9])/sprintf("%%%02X", ord($1))/seg;
```

Set-Cookie Format - Expires



Interestingly, this time value is **comparing with the browser time**, not the server time!

```
[Day of week], DD-Mon-YYYY HH:MM:SS GMT
```

```
E.g., "Sun, 28-Feb-2010 00:00:00 GMT"
```

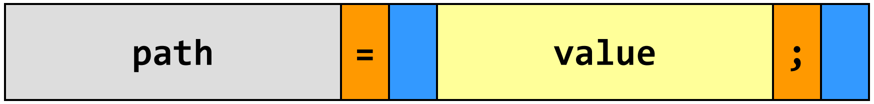
See gmtime() in http://perldoc.perl.org/

By design, HTTP cookie will expire on the client side.

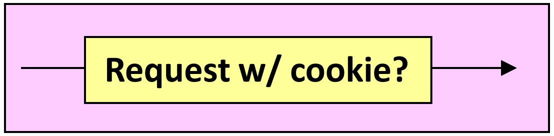
The "**expires**" field sets the time at which the cookie expires.

```
if( "expires" does not exist ), then
    the cookie expires at the end of the session, i.e,
    expires when the browser is closed.
else
    it states the exact time that the cookie expires.
```

Set-Cookie Format - Path



The "**path**" field sets the path and its sub-tree that the cookie is available.

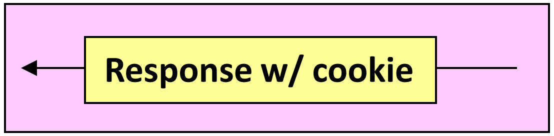


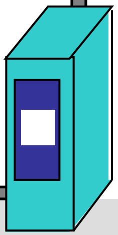
Example.

```
If "path= /~tywong/" is set, then

    only pages hosted under the path "/~tywong/"
         e.g., "/~tywong/cgi-bin/example.cgi"
    will be able to receive this cookie;
```

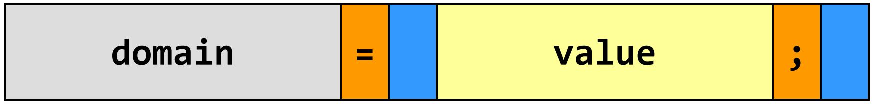
If the "**path**" field does not exist…

the pathname of the response message will be used.





Set-Cookie Format - Domain



The “**domain**” field sets the domain that the cookie is available.

Example.

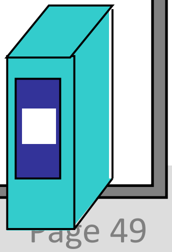
```
If “domain=.cse.cuhk.edu.hk” is set, then

websites hosted in
     “appsrv.cse.cuhk.edu.hk” and “www.cse.cuhk.edu.hk” will be
able to receive this cookie; but not “www.cse.ust.hk”.
```

Request w/ cookie?

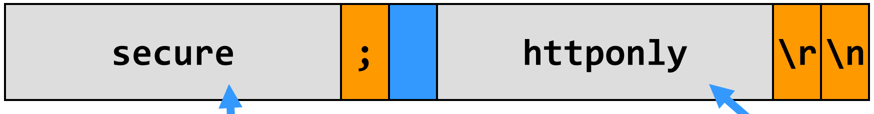
If the “**domain**” field does not exist…

the host name of the response message will be used.

Response w/ cookie



Set-Cookie Format - secure & httponly



If the "secure" field appears, then

this cookie will be available only when the connection is HTTPS, not HTTP.

If the "httponly" field appears, then

this cookie will be available only when not using JavaScript, no matter the connection is HTTP or HTTPS.

Set-Cookie Format

```
Set-Cookie: hello=world\r\n
```

```
Set-Cookie: hello=world; expires=Sun, 28-Feb-2011 00:00:00 GMT\r\n
```

```
Set-Cookie: hello=world; expires=Sun, 28-Feb-2011 00:00:00 GMT;
            domain=.cse.cuhk.edu.hk; path=/~csci4140\r\n
```

```
Set-Cookie: hello=world; expires=Sun, 28-Feb-2011 00:00:00 GMT;
            secure; httponly\r\n
```

Note that one response message can hold more than one “**`Set-Cookie`**” header.

Some examples

How about the client side?

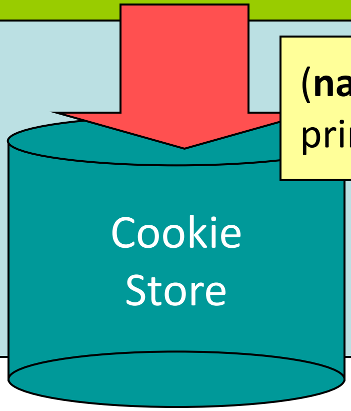
[Cookie Name]	=		value	;	
expires	=		value	;	
path	=		value	;	
domain	=		value	;	
secure	;		httponly	\r	\n

The client side must store every incoming cookie. But...

What if the **name** collides?

What if the **domain** collides?

What if the **path** collides?

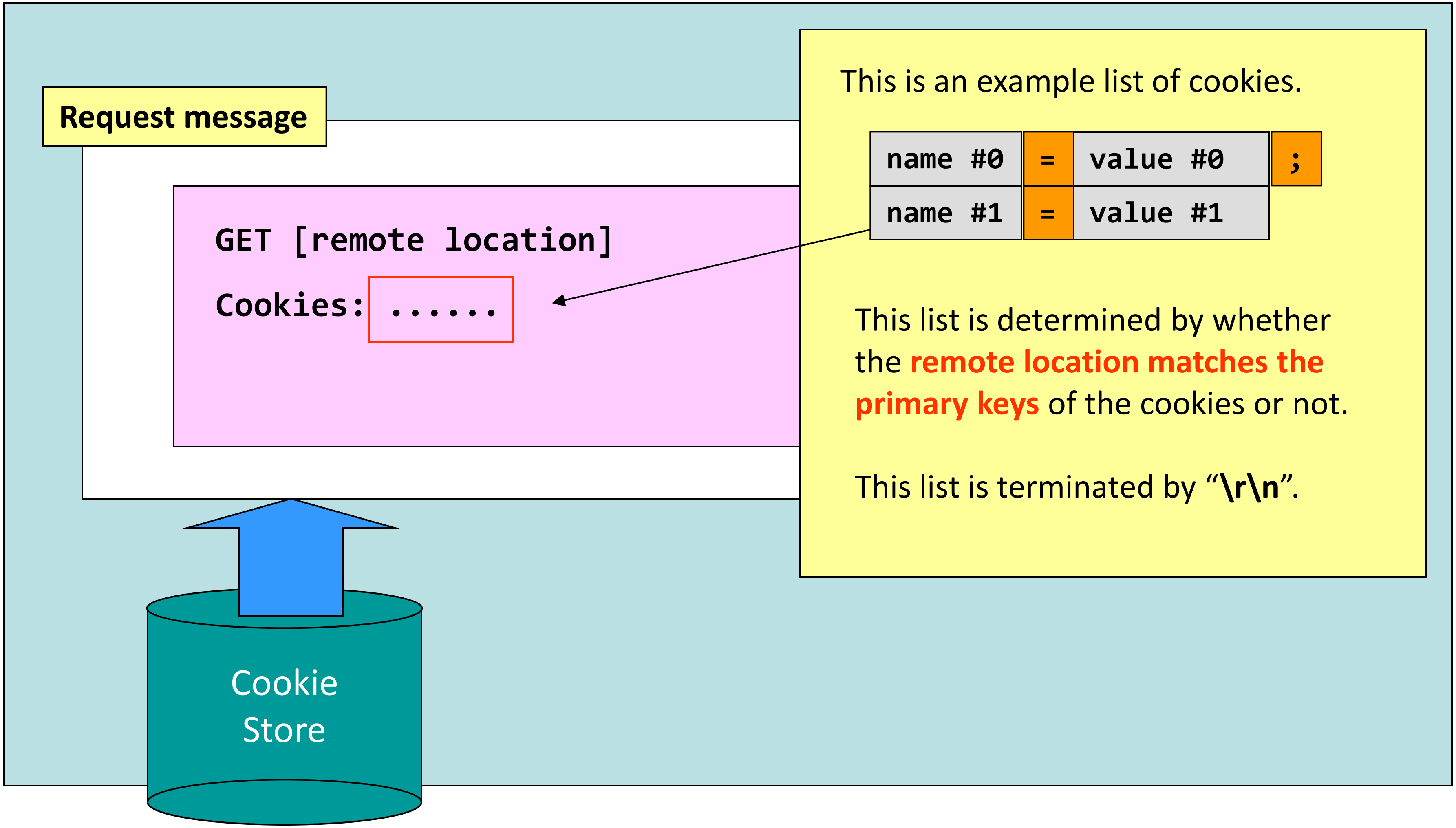


(**name, domain, path**) together form a primary key for the cookie store…

If the key can be found in the cookie store, then the old value will be replaced.

Else, a new cookie entry will be created.

How about the client side?



```
GET [remote location]
Cookies: ......
```

This is an example list of cookies.

name #0	=	value #0	;
name #1	=	value #1	

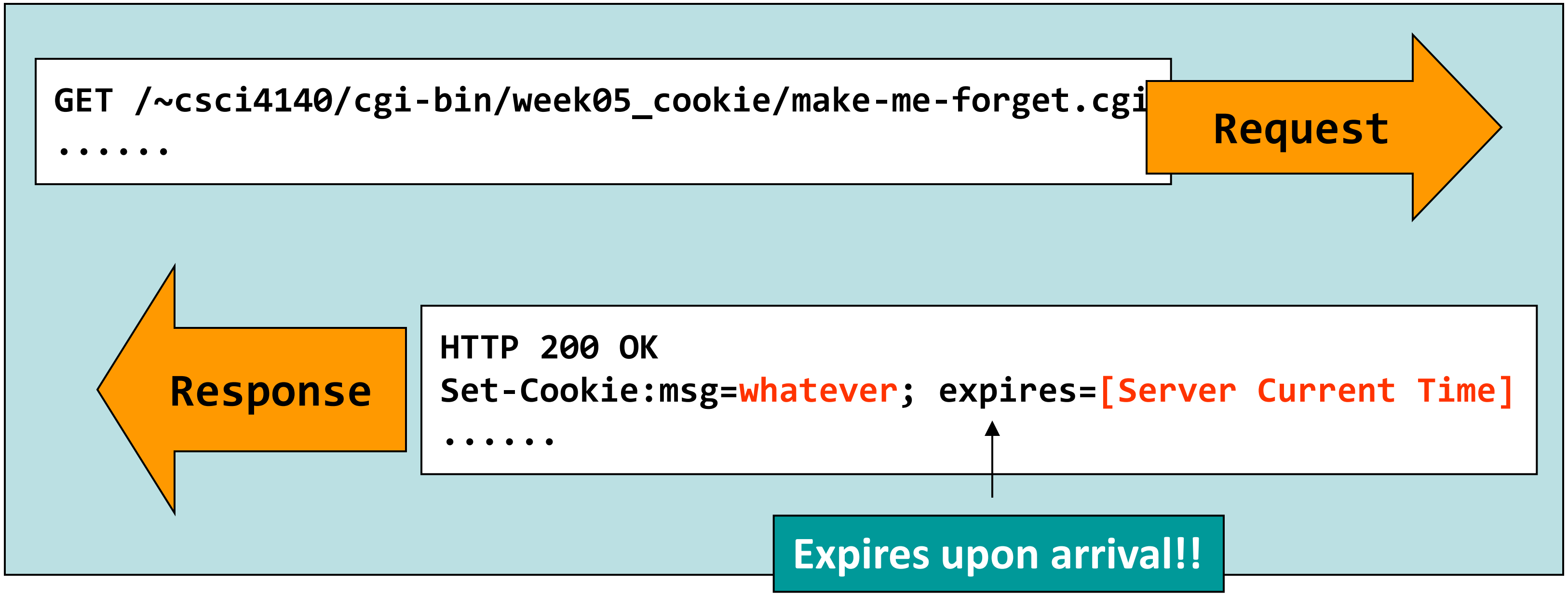
This list is determined by whether the **remote location matches the primary keys** of the cookies or not.

This list is terminated by “**\r\n**”.

[Example] “example.cgi”, “cookie_form/”, “steal_cookies.html”

Delete Cookies?

- How to make the browser to forget?
  - Waiting for expiry?
  - Removing the cookie files manually?



```
GET /~csci4140/cgi-bin/week05_cookie/make-me-forget.cgi
......
```

```
HTTP 200 OK
Set-Cookie:msg=whatever; expires=[Server Current Time]
......
```

Expires upon arrival!!

Summarize the role of cookies…

A login-based Web system needs a **session management sub-system**.

A login session has to be stored in a **permanent storage** on both the client and the server side.

A session management sub-system needs to **expire idle login sessions**.

Concerns

because HTTP is stateless…

Unless the session is only valid during the browser is running, else, this requires users to login again and this **gives troubles** to users. This may also **leaves useless session records** on the server side.

because of security and management concerns…

Reasons

Summarize the role of cookies…

Concerns

A login-based Web system needs a **session management sub-system**.

A login session has to be stored in a **permanent storage** on both the client and the server side.

A session management sub-system needs to **expire idle login sessions**.

Client side

HTTP protocol submits HTTP cookies to the server side ***together with every request***. This allows the transfer of the session key.

HTTP cookie is designed to be stored on the permanent storage of the client program.

HTTP cookie has an expiry mechanism implemented.

Summarize the role of cookies…

A login-based Web system needs a **session management sub-system**.

A login session has to be stored in a **permanent storage** on both the client and the server side.

A session management sub-system needs to **expire idle login sessions**.

Concerns

HTTP cookie can be read by CGI programs as an environment variable.

The session key has to be stored and **actively maintained** (e.g., session expiry on the server side, not the cookie) by the server.

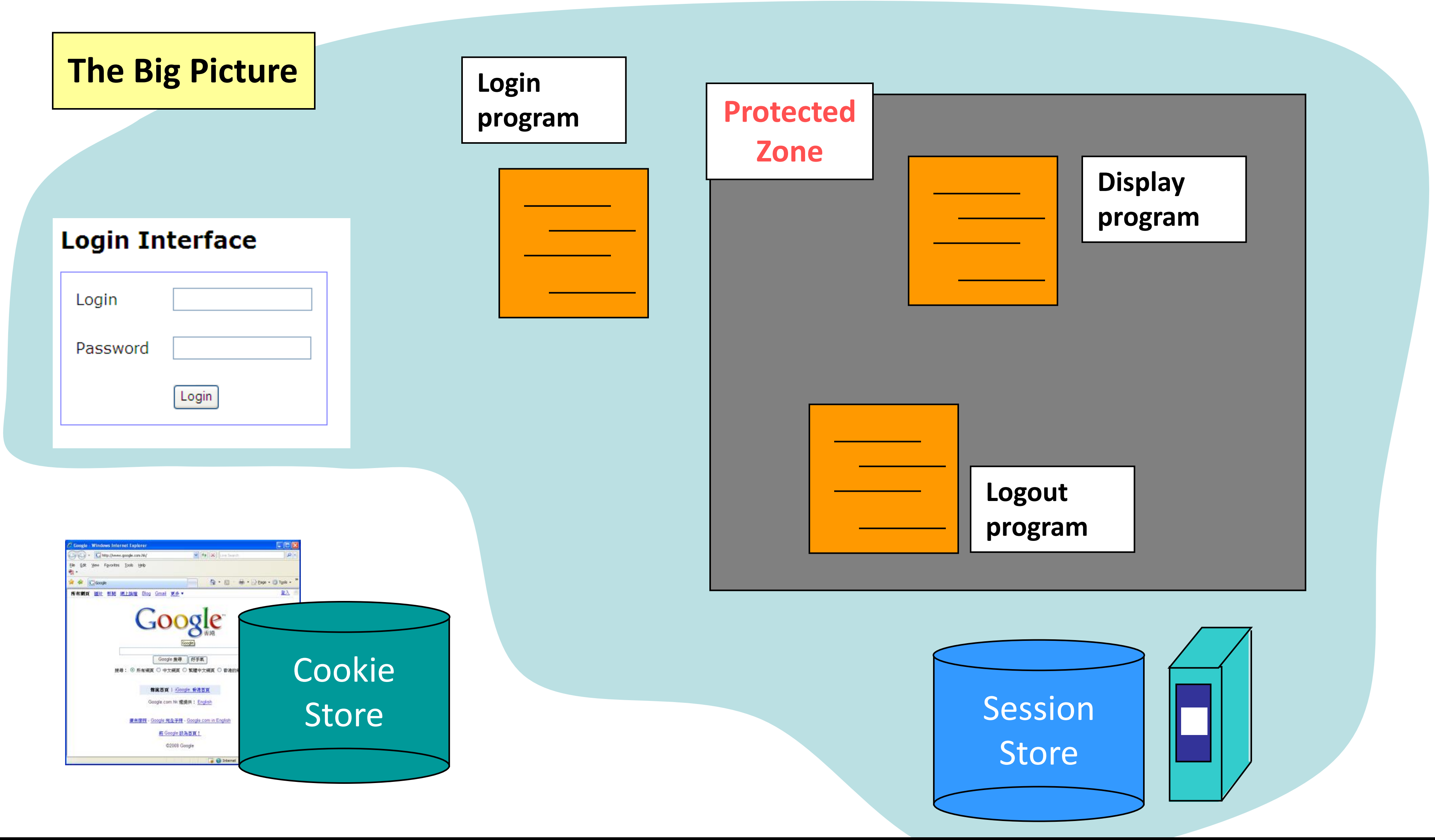
Server has to implement a mechanism to expire idle login sessions.

Server side

HTTP Cookie

– a sample system using cookie...

A Typical Web System Design…



The Big Picture

Login program

Protected Zone

Display program

Logout program

Login Interface

Login

Password

Login

Cookie Store

Session Store

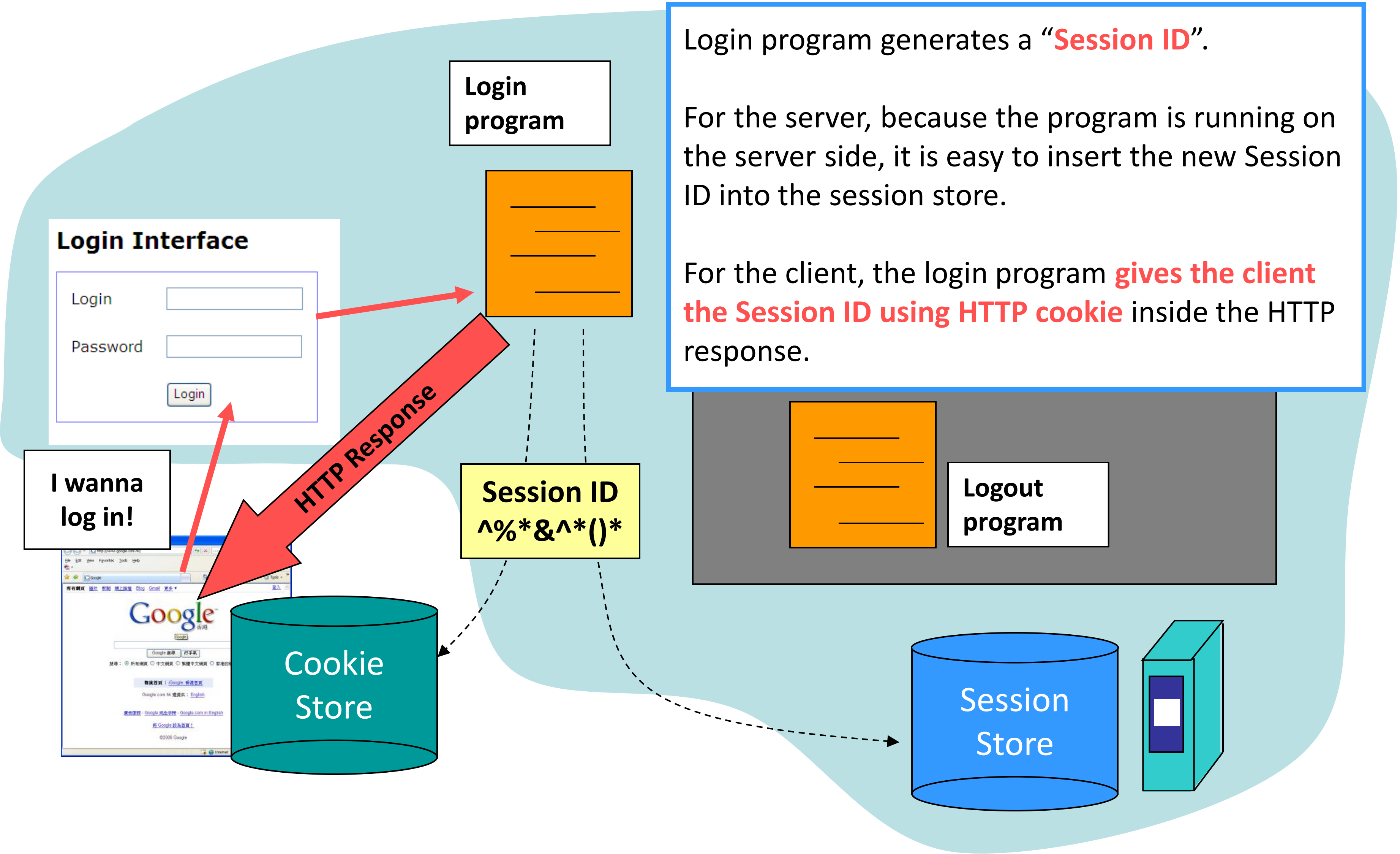
[Example] “cookie_system/”

A Typical Web System Design…

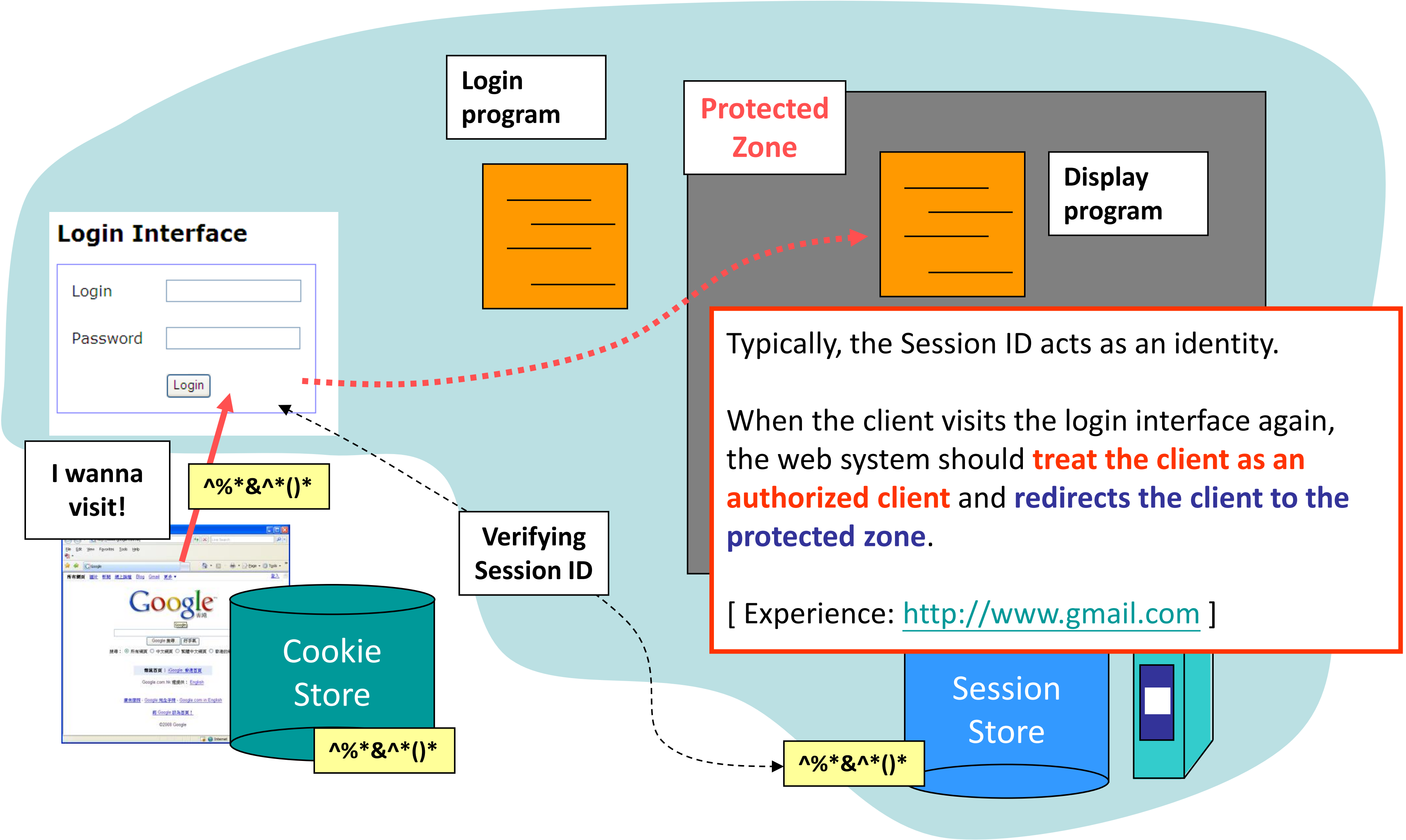
Login program generates a "**Session ID**".

For the server, because the program is running on the server side, it is easy to insert the new Session ID into the session store.

For the client, the login program **gives the client the Session ID using HTTP cookie** inside the HTTP response.



A Typical Web System Design...

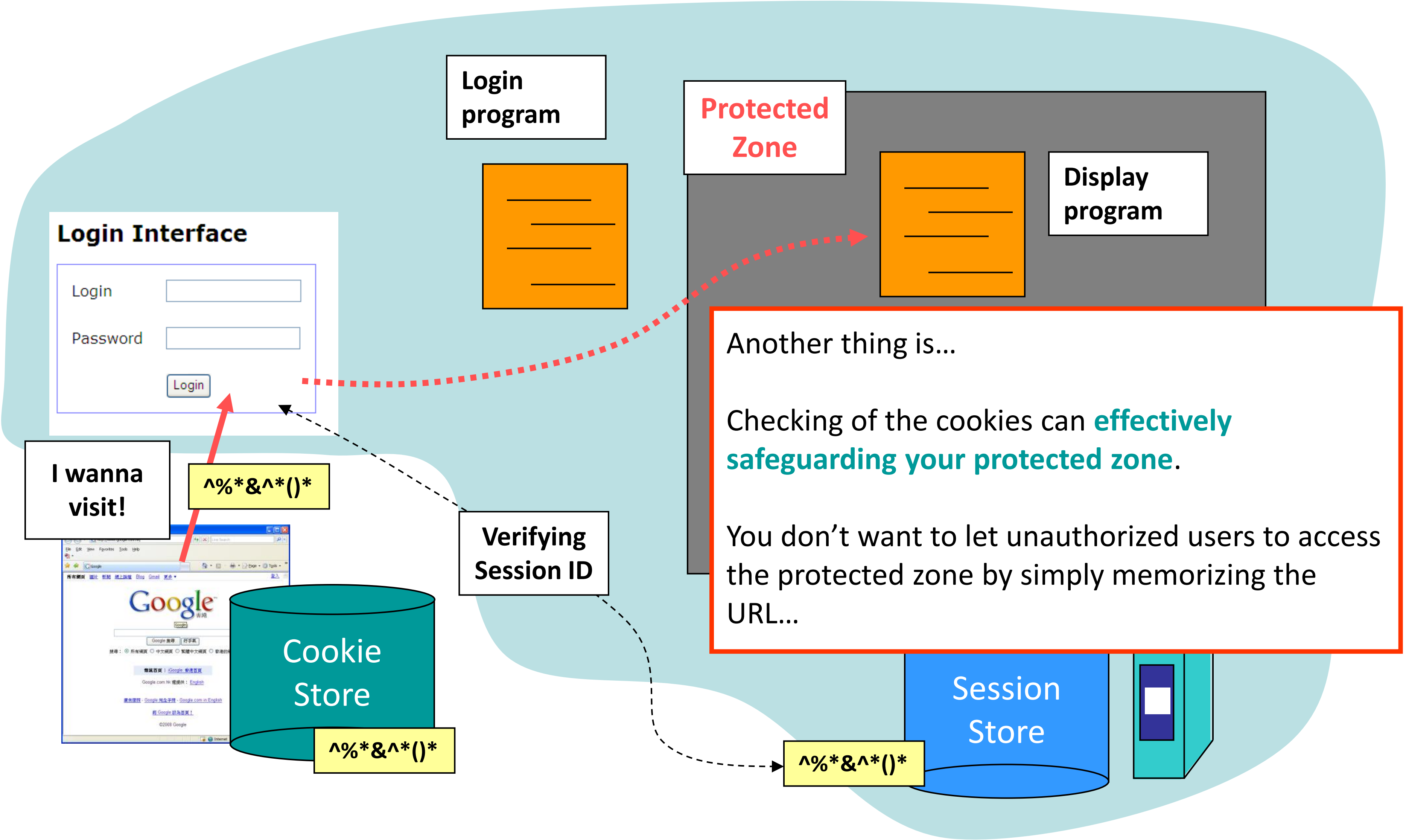


Typically, the Session ID acts as an identity.

When the client visits the login interface again, the web system should **treat the client as an authorized client** and **redirects the client to the protected zone**.

[Experience: http://www.gmail.com]

A Typical Web System Design…

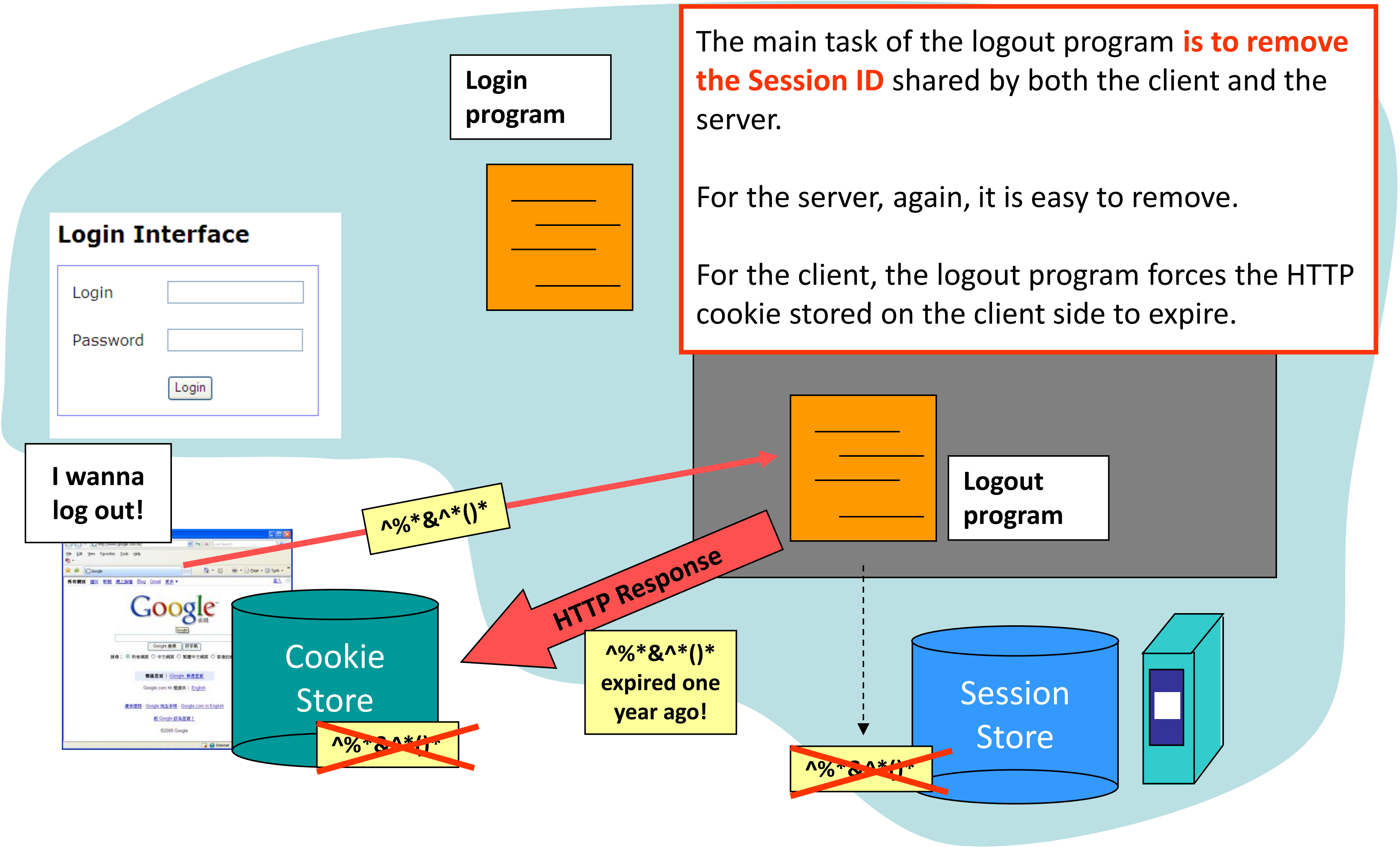


Another thing is…

Checking of the cookies can **effectively safeguarding your protected zone**.

You don’t want to let unauthorized users to access the protected zone by simply memorizing the URL…

A Typical Web System Design…



The main task of the logout program **is to remove the Session ID** shared by both the client and the server.

For the server, again, it is easy to remove.

For the client, the logout program forces the HTTP cookie stored on the client side to expire.

Persistent HTTP Cookie

– the best practice should be used ...

Persistent cookie

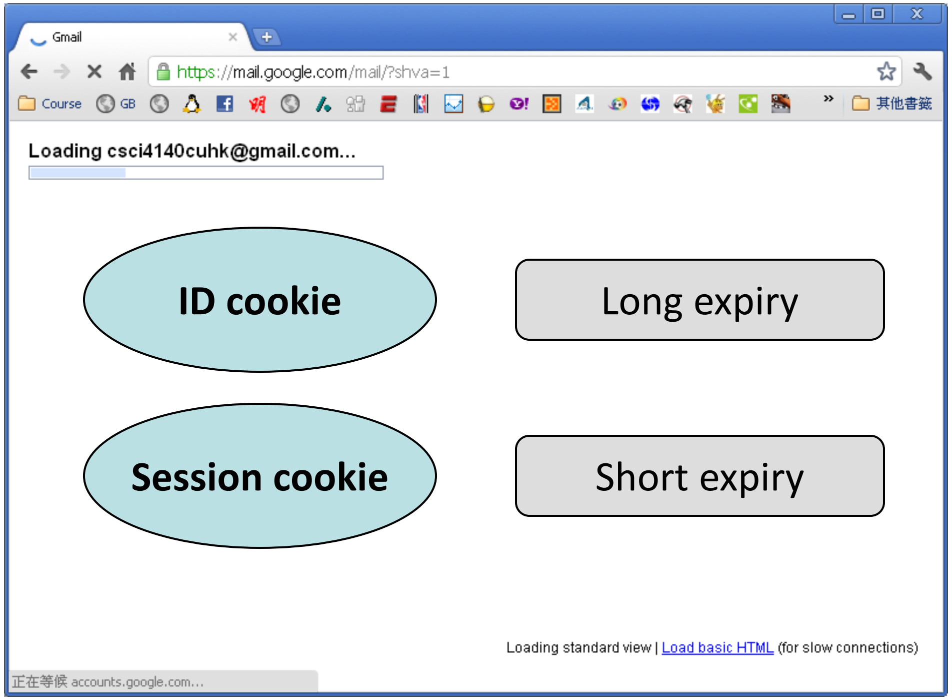
- Basically, using a persistent cookie introduces security problems:
  - Timeout? Until 2036?
  - Replay attack?

- Let’s assume the following:
  - Cookie can easily be stolen;
  - Persistent session cookie is equivalent to your credentials.
  - A user may wish to have persistent cookies on multiple web browsers on different machines simultaneously.

- Goal:
  - Minimize the damages caused by cookie thefts.

Persistent cookie – best practice

- Best effort in detecting abnormal situation.



Loading csci4140cuhk@gmail.com...

Loading standard view | Load basic HTML (for slow connections)

正在等候 accounts.google.com...

ID cookie — Long expiry

Session cookie — Short expiry

We understand that after each successful login attempt, a session cookie should be set.

Yet, we can generate another cookie as the identifier of the entire persistent session.

A New Cookie!

Persistent cookie – best practice

- Best effort in detecting abnormal situation.

username	varchar (100)	PRIMARY 1	plaintext
password	varchar (32)		MD5 hashed
id_cookie	varchar (32)	PRIMARY 2	MD5 hashed
session_cookie	varchar (32)		MD5 hashed

```
$id_cookie = md5( $username + rand() );
```

Since the username is less sensitive, so…

Later, it will shine!

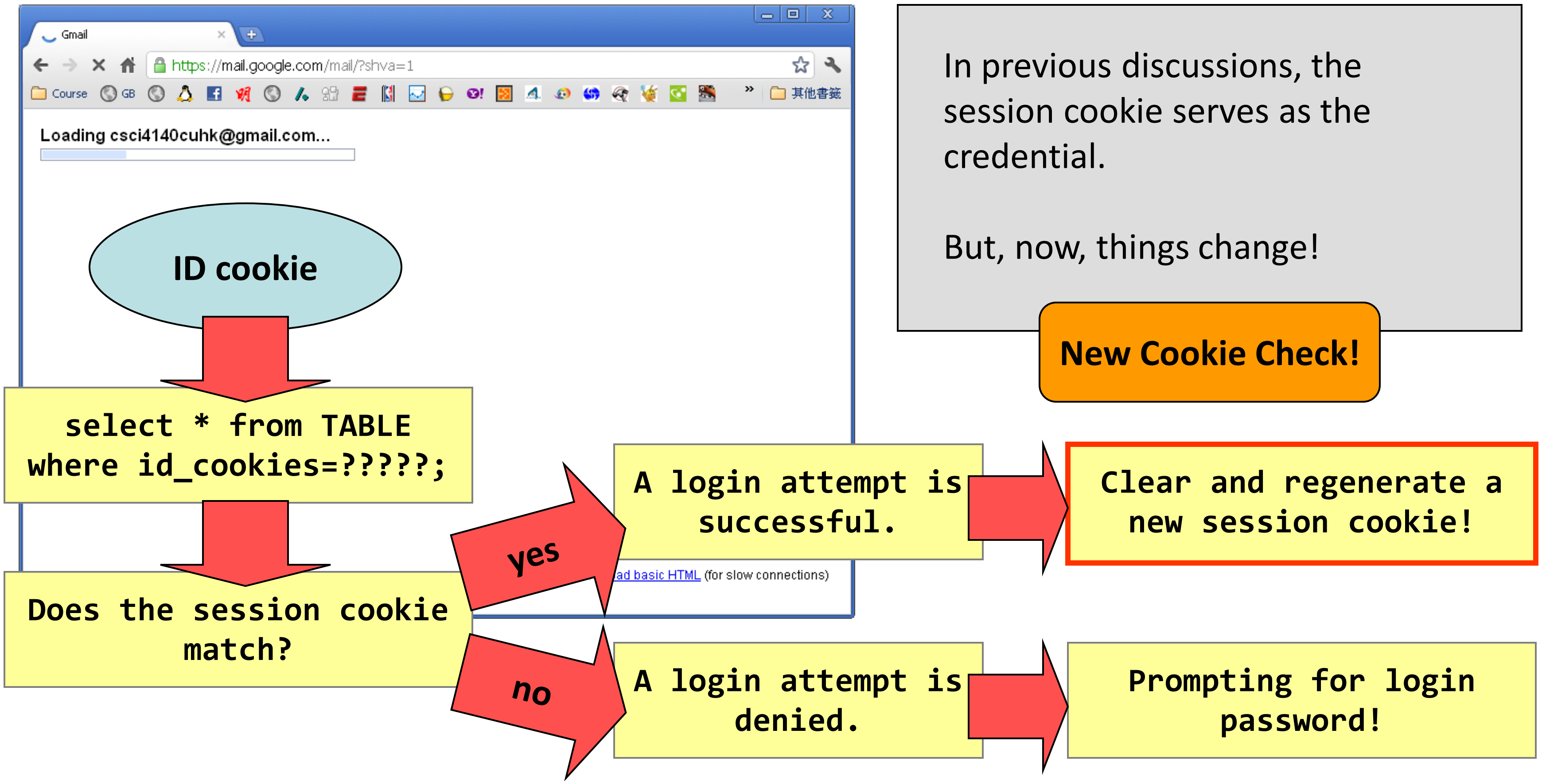
```
$session_cookie = md5( rand() );
```

A random string is a nice value!

This cookie is for the user to navigate the protected zone.

Persistent cookie – best practice

- When the login page is visited again...


Gmail
https://mail.google.com/mail/?shva=1
Course
GB
其他書籤
Loading csci4140cuhk@gmail.com...
ad basic HTML (for slow connections)
yes
no

In previous discussions, the session cookie serves as the credential.

But, now, things change!

New Cookie Check!

ID cookie

```
select * from TABLE
where id_cookies=?????;
```

Does the session cookie match?

A login attempt is successful.

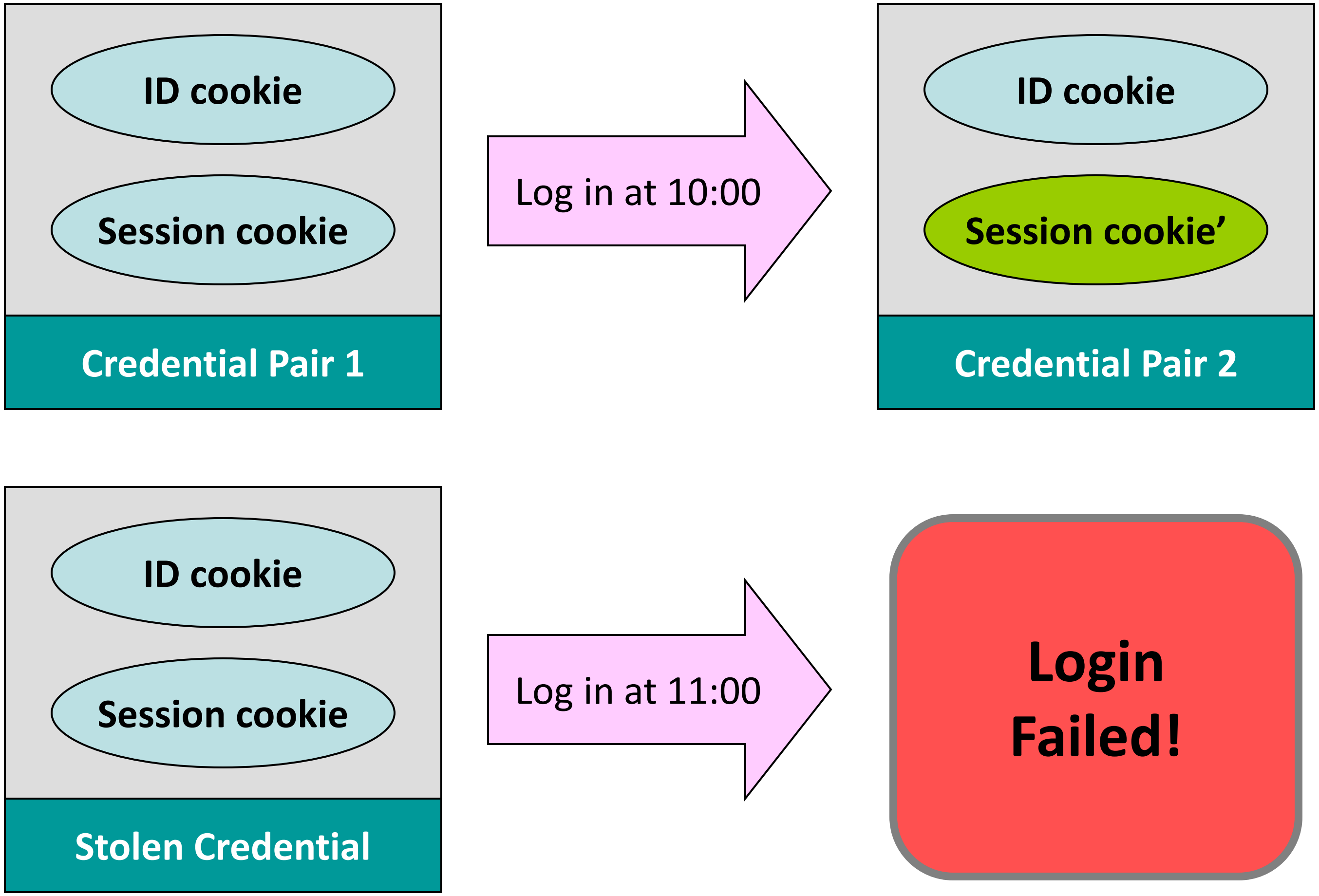
Clear and regenerate a new session cookie!

A login attempt is denied.

Prompting for login password!

Persistent cookie – best practice

- I can't catch the whole picture!!!



Persistent cookie – best practice

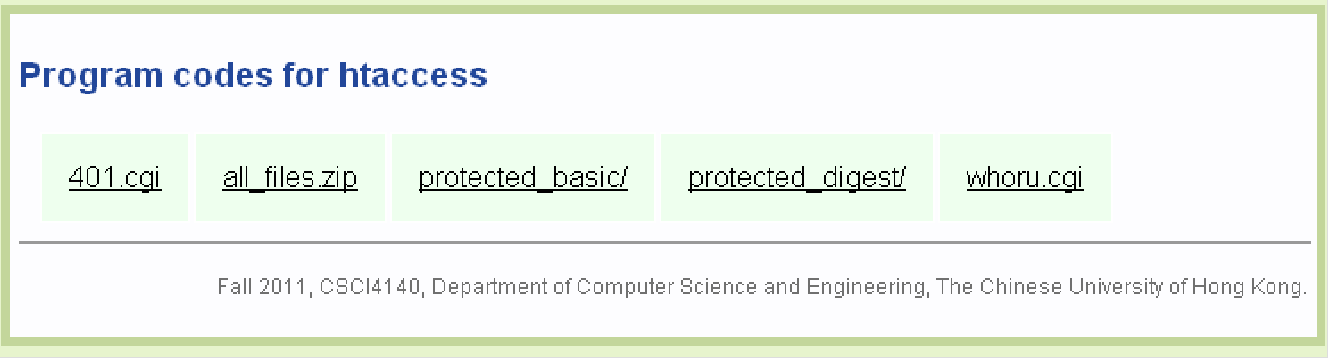
- The best practice is also a best effort implementation.
  - The weakness lies in the timing of the attack.
  - At least, the system can detect and flag out any abnormal login attempts.

- The best part of this approach:

```
$id_cookie = md5( $username + rand() );
```

It allows a user to have multiple login sessions in multiple browsers on different machines.

Web Server’s Authentication



Program codes for htaccess

401.cgi | all_files.zip | protected_basic/ | protected_digest/ | whoru.cgi

Fall 2011, CSCI4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

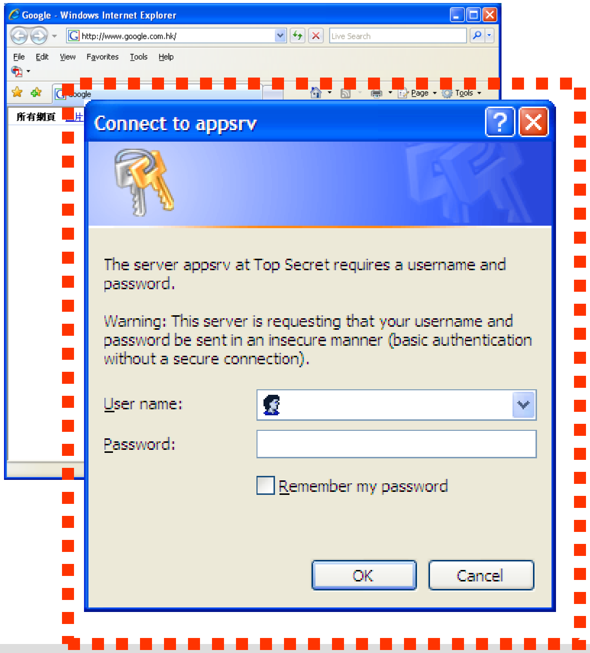
http://demo4140-tywong.rhcloud.com/05_htaccess/

Log in Using…

HTTP Authentication.

Mechanism (1) – HTTP Authentication

- From a user perspective, HTTP authentication is…


Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
Page
Tools
所有網頁

Connect to appsrv

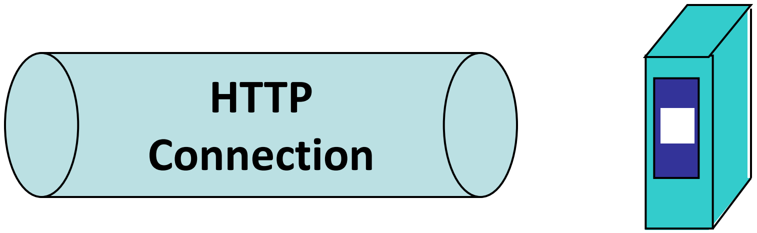
The server appsrv at Top Secret requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name:

Password:

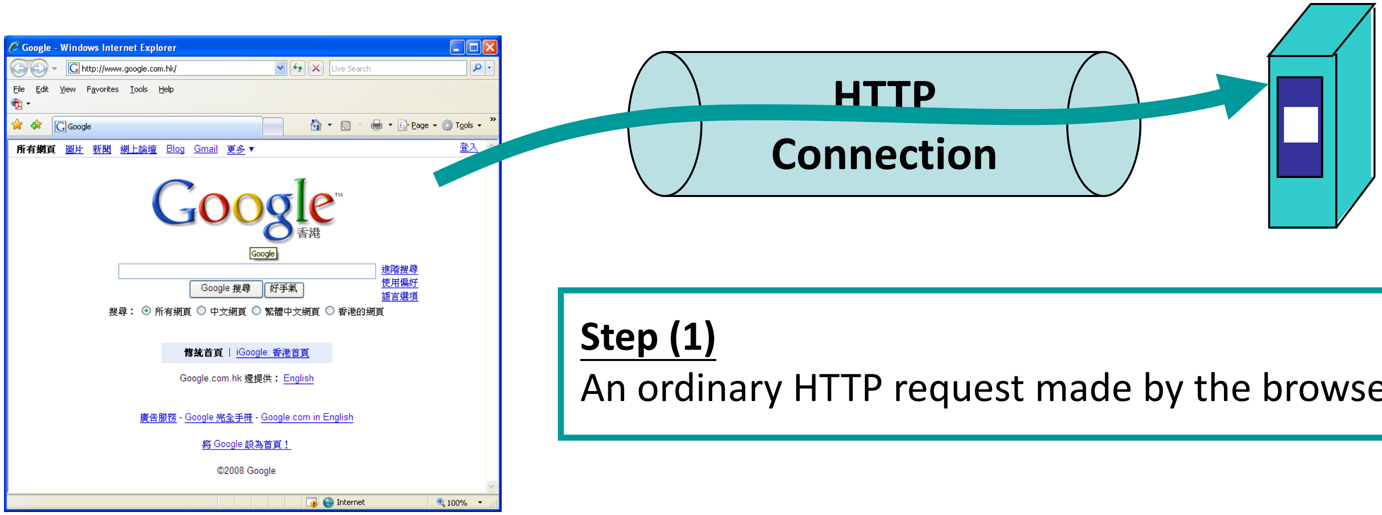
□ Remember my password



The login dialog box!

Mechanism (1) – HTTP Authentication

- From a system perspective, HTTP authentication is **access control**…

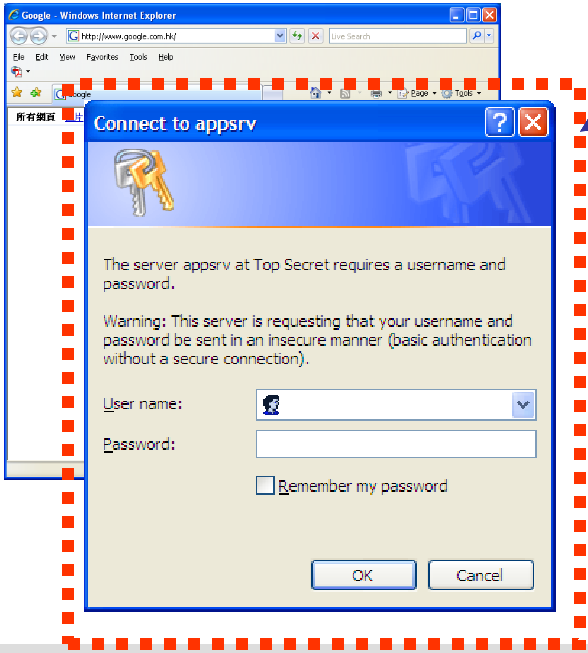


Step (1)

An ordinary HTTP request made by the browser.

Mechanism (1) – HTTP Authentication

- From a system perspective, HTTP authentication is **access control**…



Connect to appsrv

The server appsrv at Top Secret requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

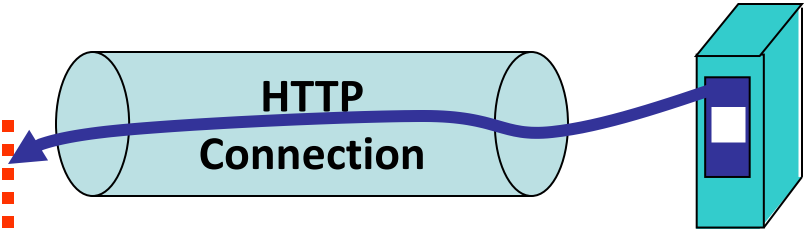
User name:

Password:

□ Remember my password

OK Cancel

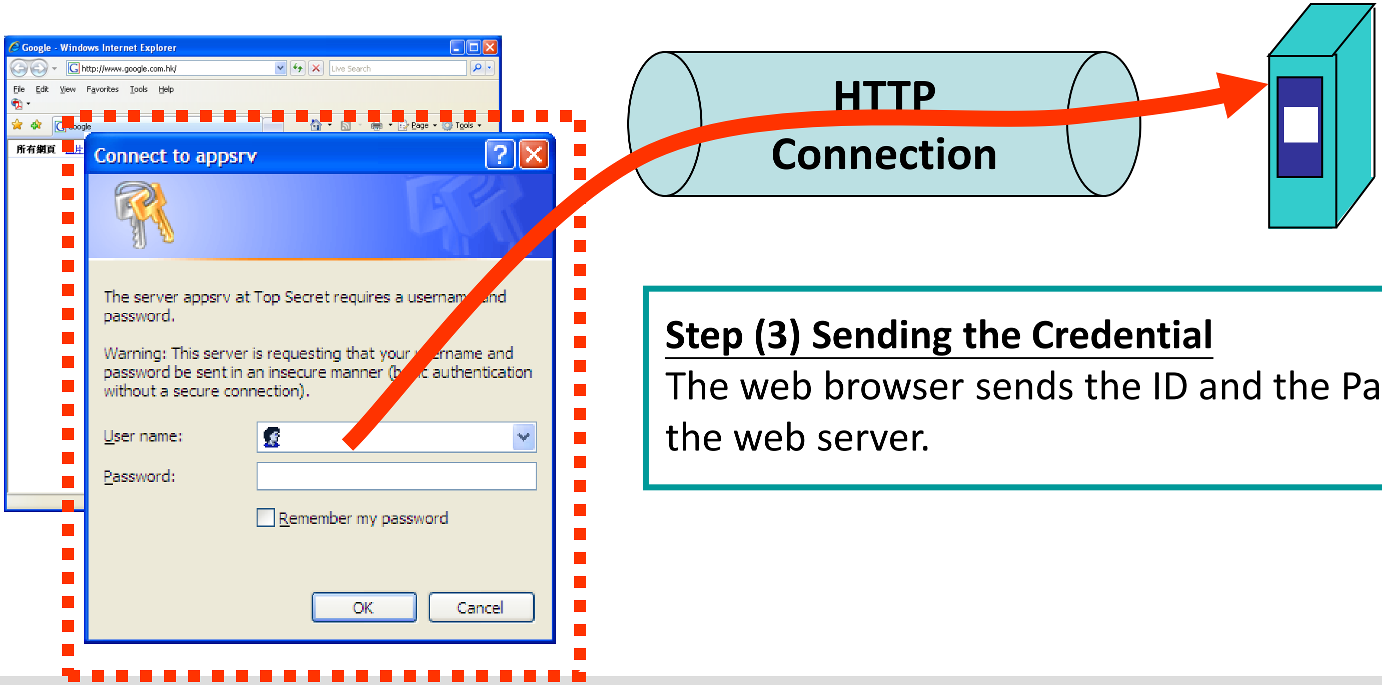
Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Page Tools
所有網頁



Step (2) 401 Unauthorized
The web server denies the browser's attempt to access a restricted area.

Mechanism (1) – HTTP Authentication

- From a system perspective, HTTP authentication is **access control**…



Google - Windows Internet Explorer
http://www.google.com.hk/
File Edit View Favorites Tools Help
HTTP
Connection

Connect to appsrv

The server appsrv at Top Secret requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name:

Password:

□ Remember my password

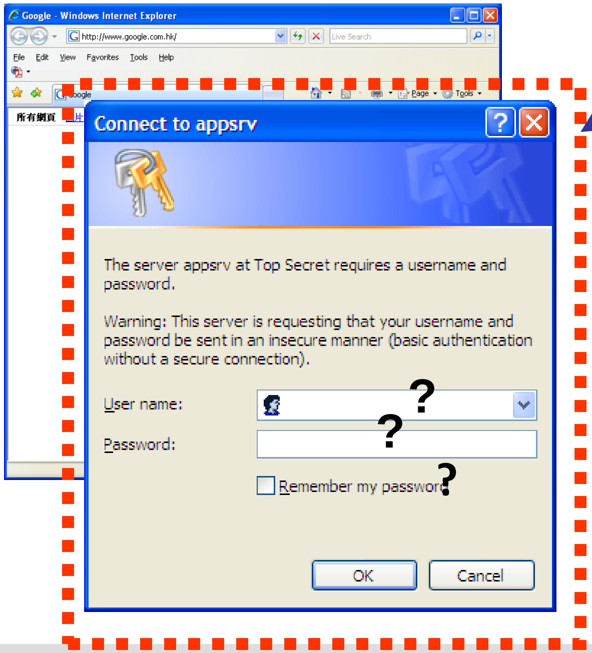
OK Cancel

Step (3) Sending the Credential

The web browser sends the ID and the Password to the web server.

Mechanism (1) – HTTP Authentication

- From a system perspective, HTTP authentication is **access control**…



Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Page ▾ Tools ▾
所有網頁

Connect to appsrv

The server appsrv at Top Secret requires a username and password.

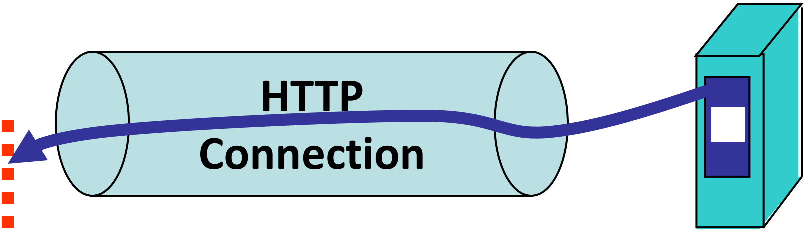
Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name: ?

Password: ?

□ Remember my password ?

OK Cancel

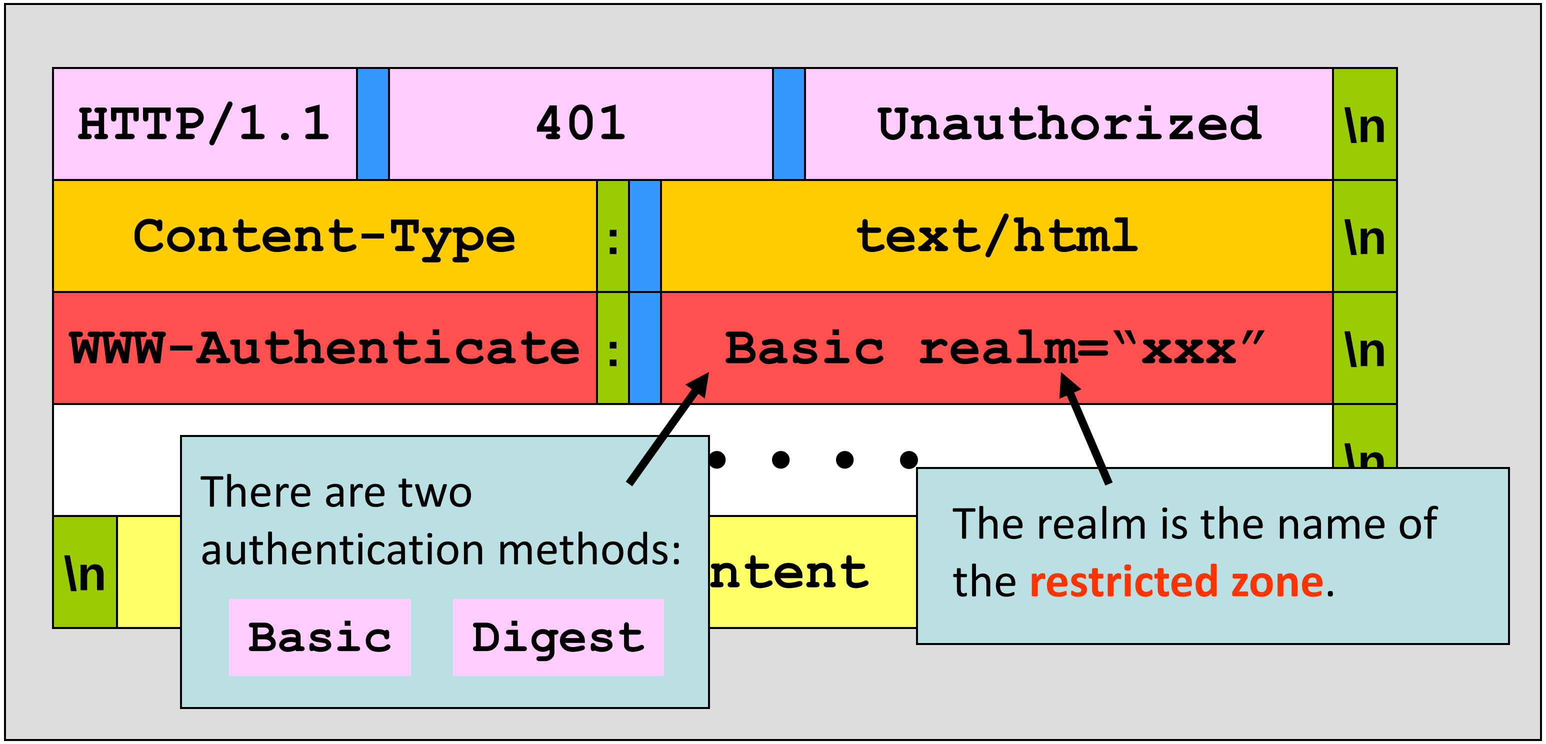


Step (4) 200 or 401?

Depending on the correctness of the credential, the web server makes different replies.

- If correct, then the reply is "**200 OK**".
- Else, the reply is "**401 Unauthorized**".

What is “`401 Unauthorized`”?



`HTTP/1.1` `401` `Unauthorized` \n

`Content-Type` : `text/html` \n

`WWW-Authenticate` : `Basic realm=“xxx”` \n

• • • • \n

\n `ntent`

There are two authentication methods:

`Basic` `Digest`

The realm is the name of the **restricted zone**.

A Server Response

Popping a “Login Dialog” Out?

```
#!/usr/local/bin/perl5.8

print  <<__MESSAGE__;
Status: 401 Unauthorized
WWW-Authenticate: Basic realm="Top Secret"
Content-type: text/html

<h1>You can never log in successfully. Wah ha ha ...</h1>

__MESSAGE__
```

To change the status code from the default “**200 OK**” to “**401 ……**”.

Message shown when the client refuses to log in.

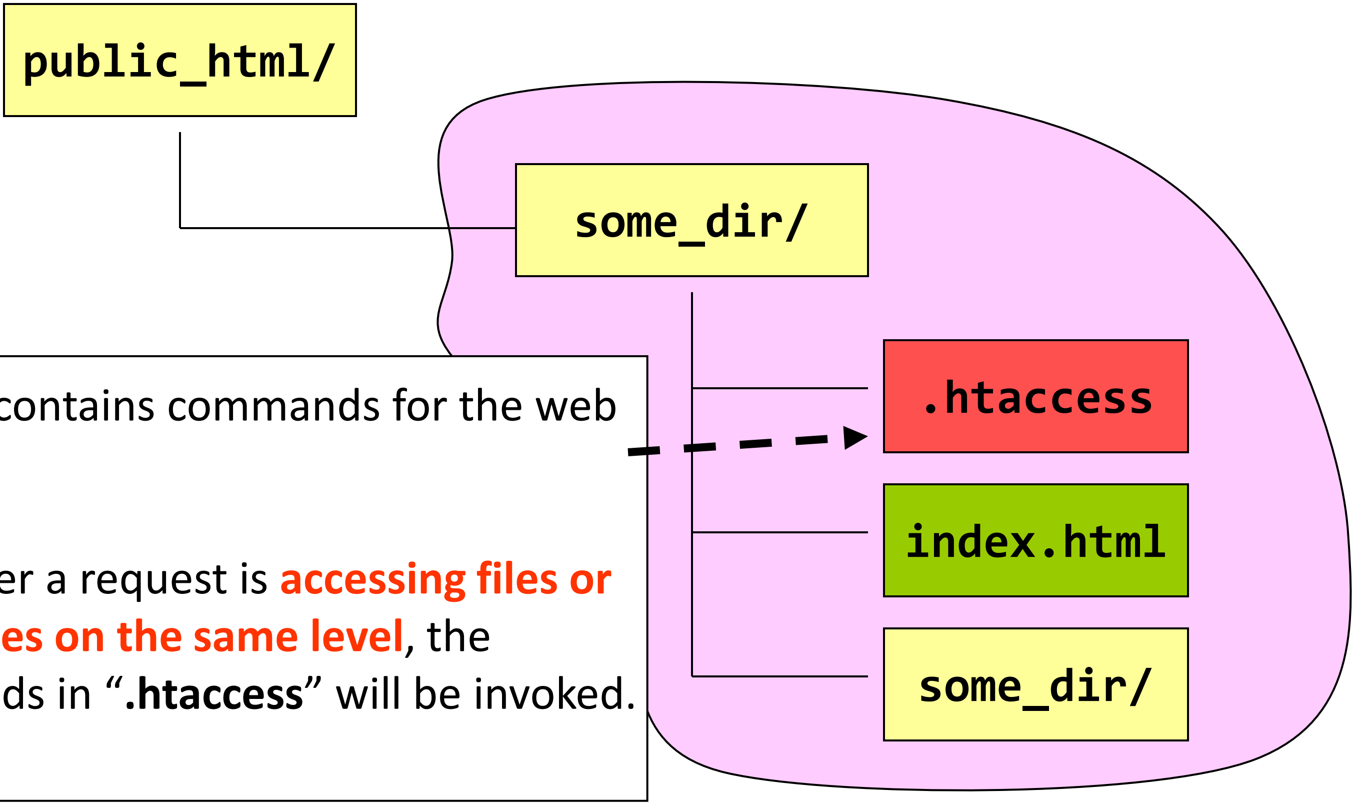
Encore for one more time!

“There’s more than one way to do it!” Perl motto!

[Example] “401.cgi”

Normal way to have the login dialog…

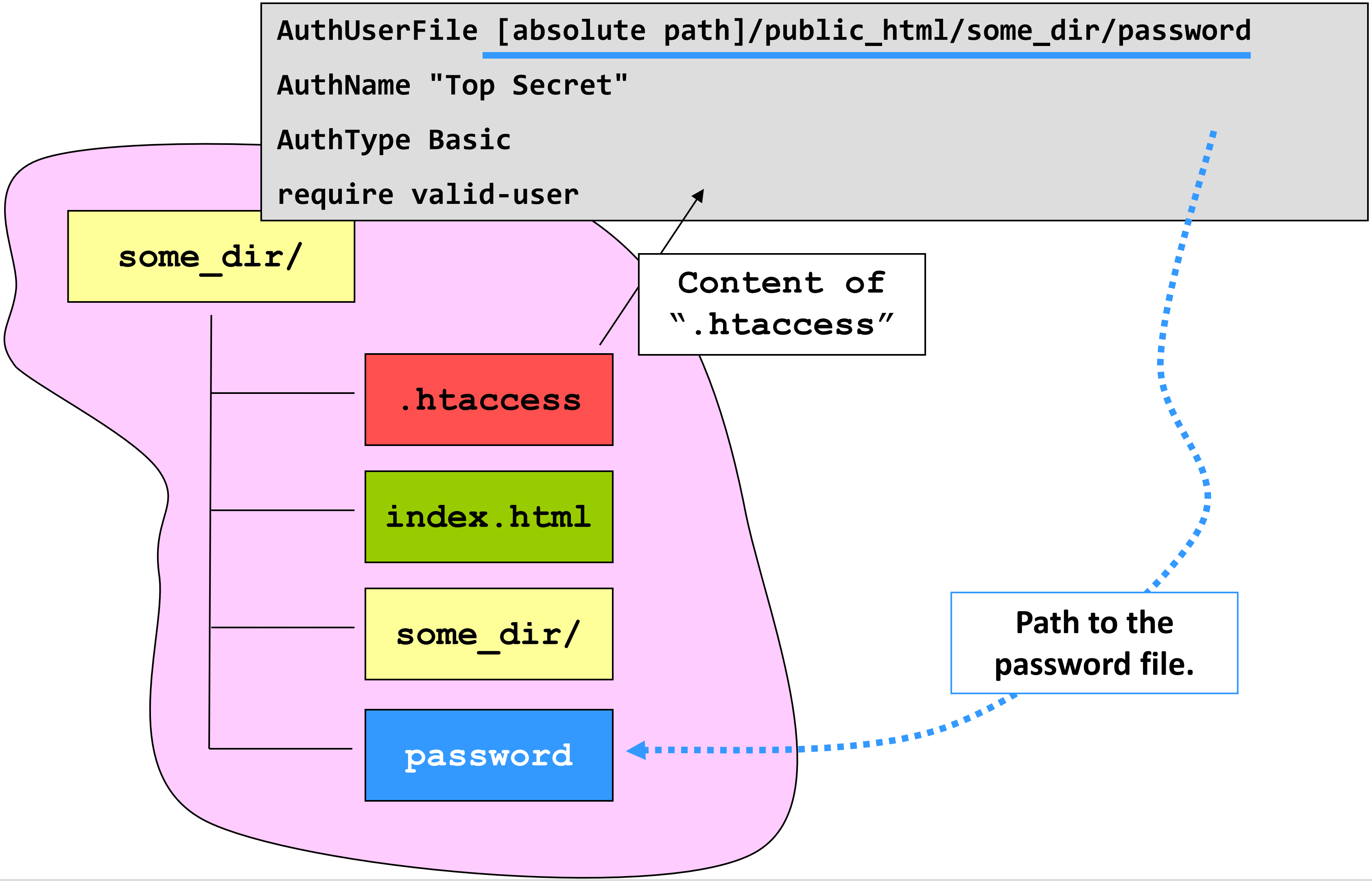
- Assume that “**`public_html`**” is the directory that hosts the htdocs…



This file contains commands for the web server.

Whenever a request is **accessing files or directories on the same level**, the commands in “**.htaccess**” will be invoked.

Normal way to have the login dialog…

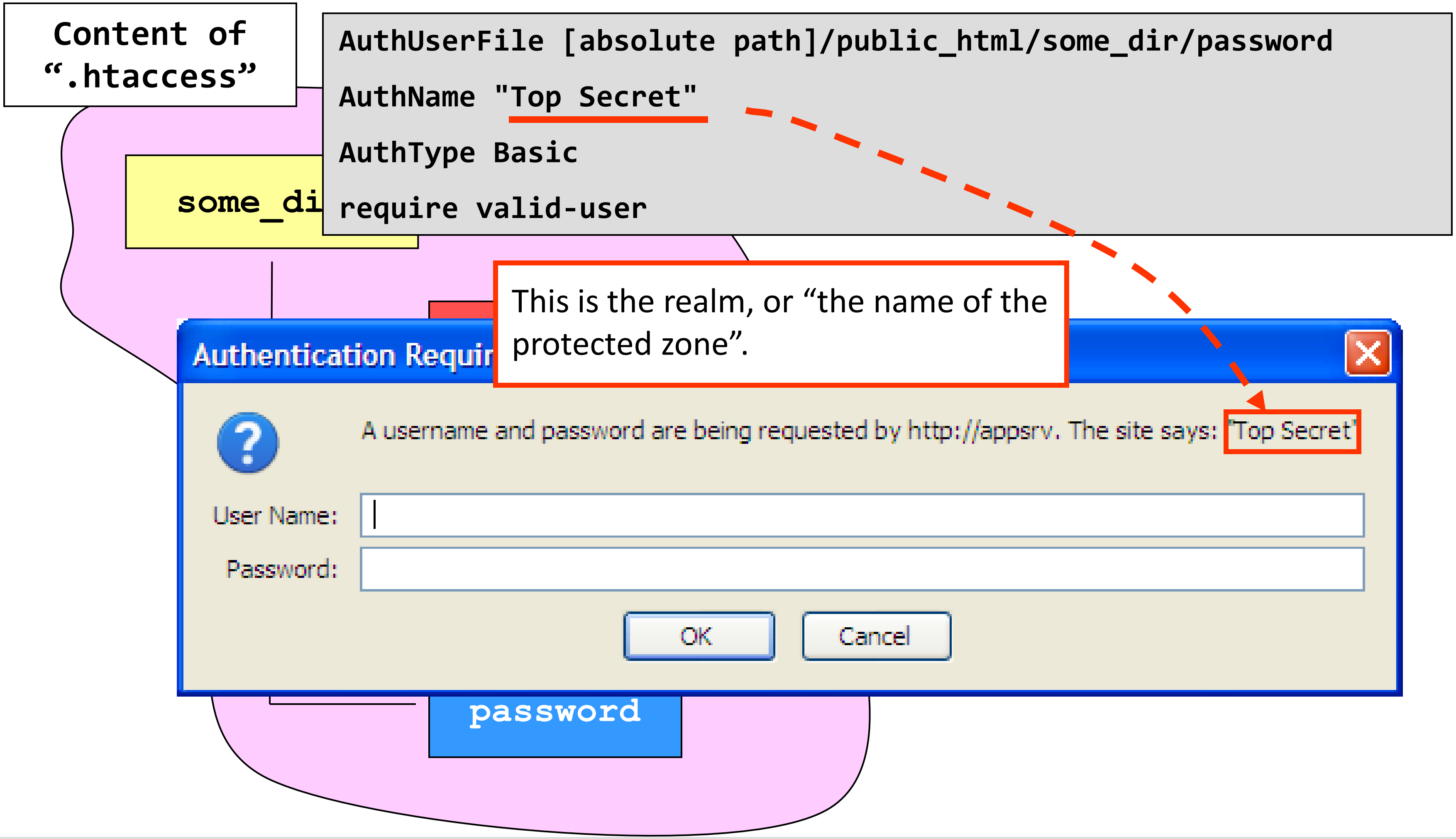


```
AuthUserFile [absolute path]/public_html/some_dir/password
AuthName "Top Secret"
AuthType Basic
require valid-user
```

Content of ".htaccess"

Path to the password file.

Normal way to have the login dialog…



Content of “.htaccess”

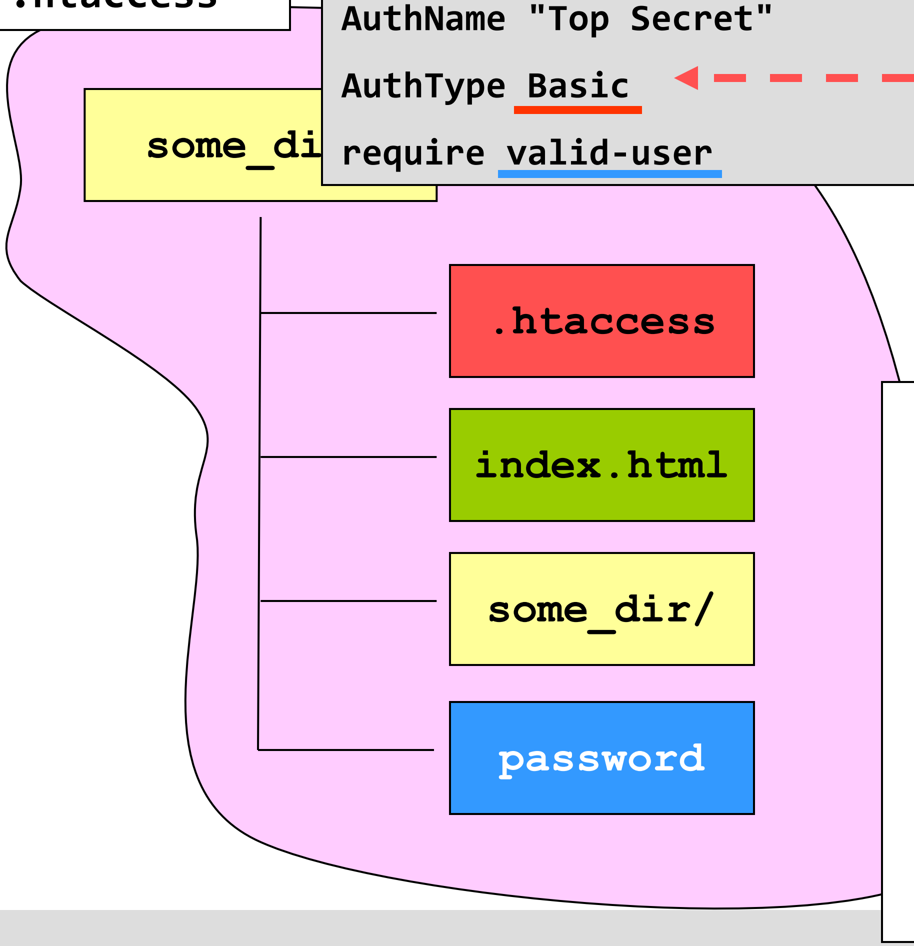
```
AuthUserFile [absolute path]/public_html/some_dir/password
AuthName "Top Secret"
AuthType Basic
require valid-user
```

This is the realm, or “the name of the protected zone”.

Authentication Requir

A username and password are being requested by http://appsrv. The site says: "Top Secret"

Normal way to have the login dialog...



Content of “.htaccess”

```
AuthUserFile [absolute path]/public_html/some_dir/password
AuthName "Top Secret"
AuthType Basic
require valid-user
```

There are two authentication methods:

Basic **Digest**

To allow all the users inside the password file to access.

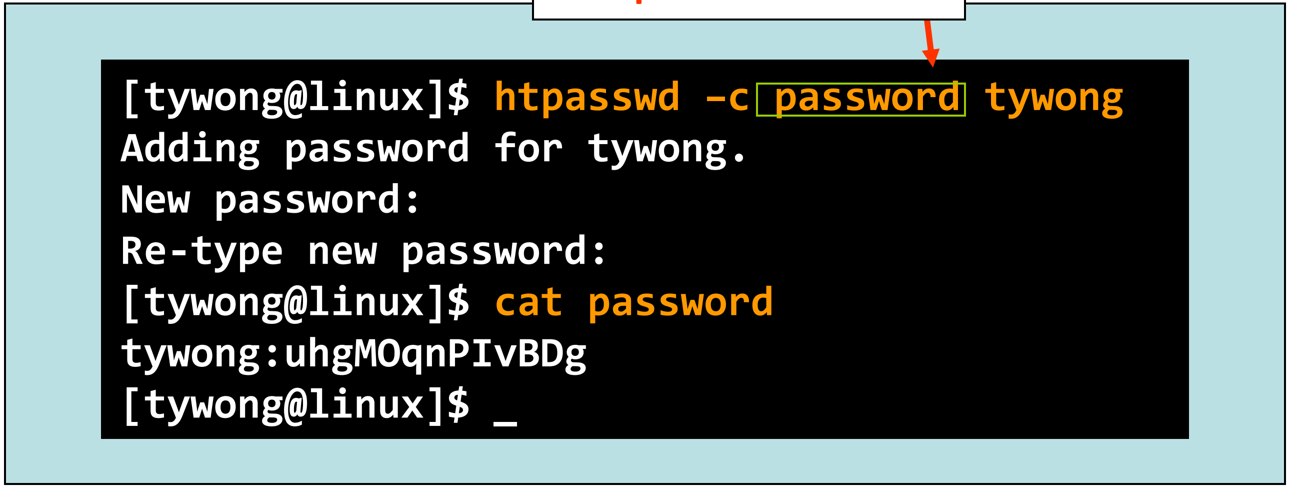
```
require user “tywong”
```

This allows “**tywong**” to log in only.

Normal way to have the login dialog...

```
AuthUserFile [absolute path]/public_html/some_dir/password
AuthName "Top Secret"
AuthType Basic
require valid-user
```

Name the password file “**password**”.



```
[tywong@linux]$ htpasswd -c password tywong
Adding password for tywong.
New password:
Re-type new password:
[tywong@linux]$ cat password
tywong:uhgMOqnPIvBDg
[tywong@linux]$ _
```

[Example] “protected_basic/password.txt”

After logged in…

```
#!/usr/bin/perl -w

print "Content-type: text/plain\n\n";

$user = $ENV{"REMOTE_USER"};
if($ENV{"REMOTE_USER"}) {
        print "You are \"$user\".\n";
} else {
        print "I don't know who you are.\n";
}
```

This variable:

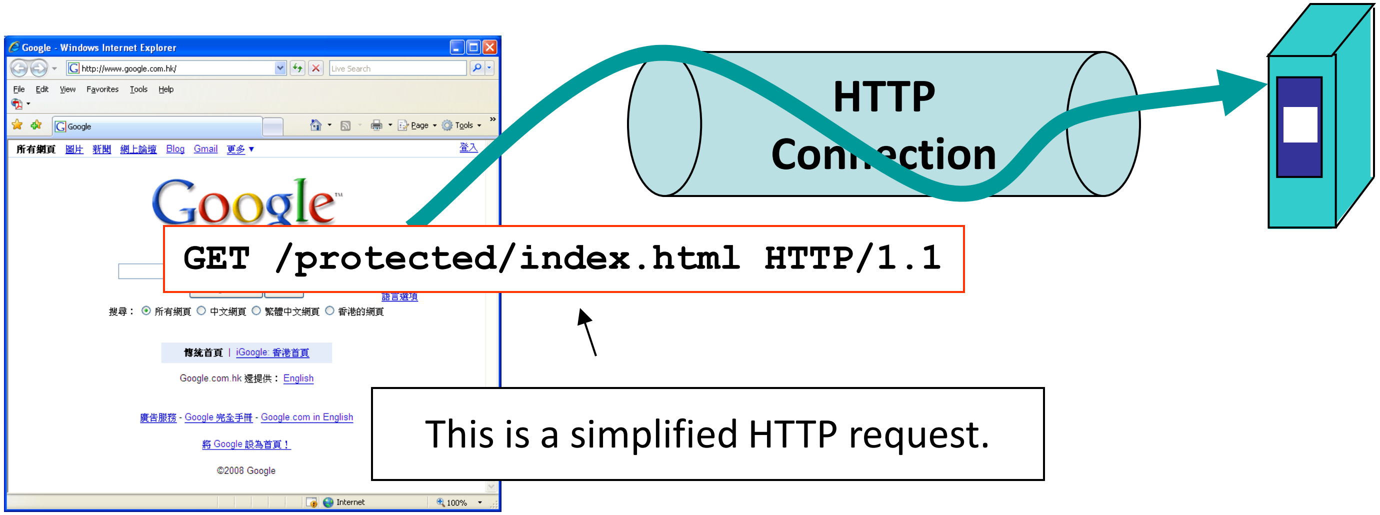
- is created by the web server, not the web browser;

- stores the identity of the allowed user.

[Example] “protected_basic/whoru.cgi”

After logged in…

- Let me ask… "do you need to log in when you access the protected page again?"



```
GET /protected/index.html HTTP/1.1
```

This is a simplified HTTP request.

Let's study what is happening during the login process…

After logged in…

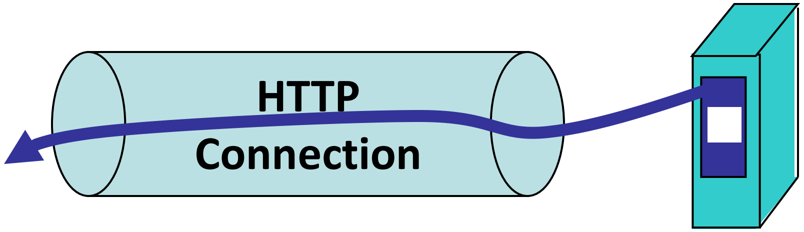
- Let me ask… "do you need to log in when you access the protected page again?"

Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
Page Tools
所有網頁 圖片 新聞 網上論壇 Blog Gmail 更多 ▼
登入
Google 香港
Google
Google 搜尋
好手氣
進階搜尋
使用偏好
語言選項
傳統首頁 | iGoogle: 香港首頁
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
Internet
100%

搜尋： ● 所有網頁 ○ 中文網頁 ○ 繁體中文網頁 ○ 香港的網頁

Google.com.hk 還提供： English

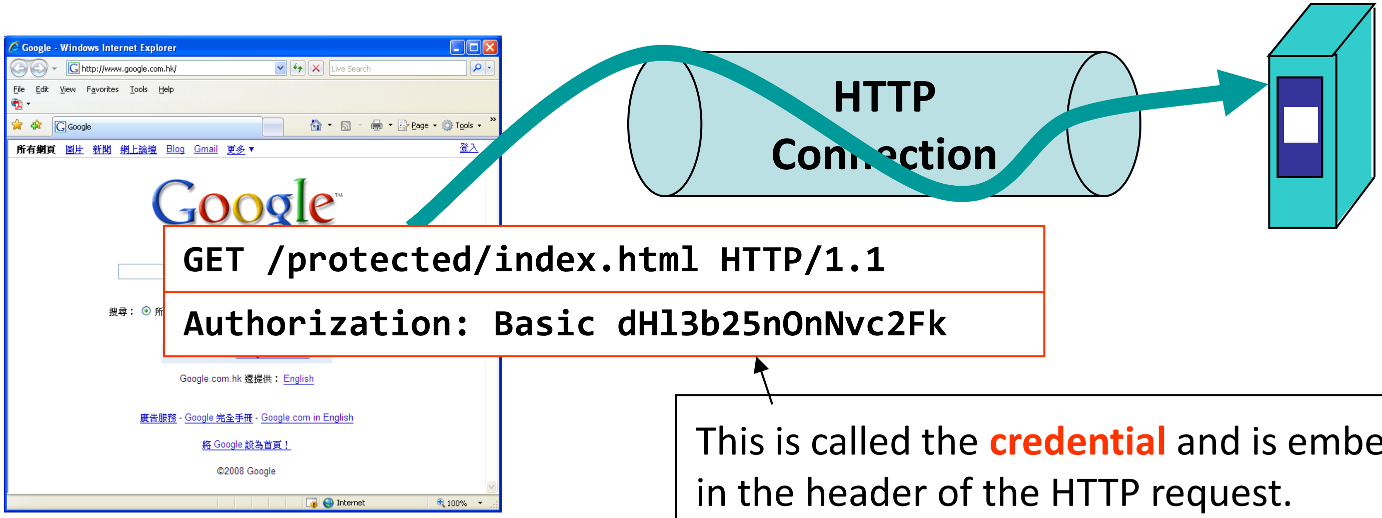
©2008 Google



```
401 Unauthorized
WWW-Authorization: Basic Realm="Top Secret"
```

After logged in…

- Let me ask… “do you need to log in when you access the protected page again?”



```
GET /protected/index.html HTTP/1.1
Authorization: Basic dHl3b25nOnNvc2Fk
```

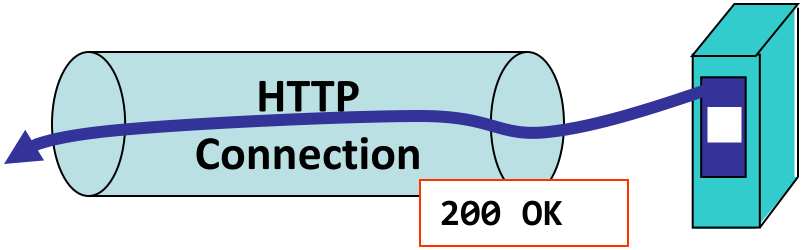
This is called the **credential** and is embedded in the header of the HTTP request.

Want the proof? Try using **Wireshark**!

After logged in…

- Let me ask… “do you need to log in when you access the protected page again?”

Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
Page ▾ Tools ▾
所有網頁 圖片 新聞 網上論壇 Blog Gmail 更多 ▾
登入
Google 香港
Google
進階搜尋
使用偏好
語言選項
Google 搜尋
好手氣
搜尋： ● 所有網頁 ○ 中文網頁 ○ 繁體中文網頁 ○ 香港的網頁
傳統首頁 | iGoogle: 香港首頁
Google.com.hk 還提供： English
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
©2008 Google
Internet
100%

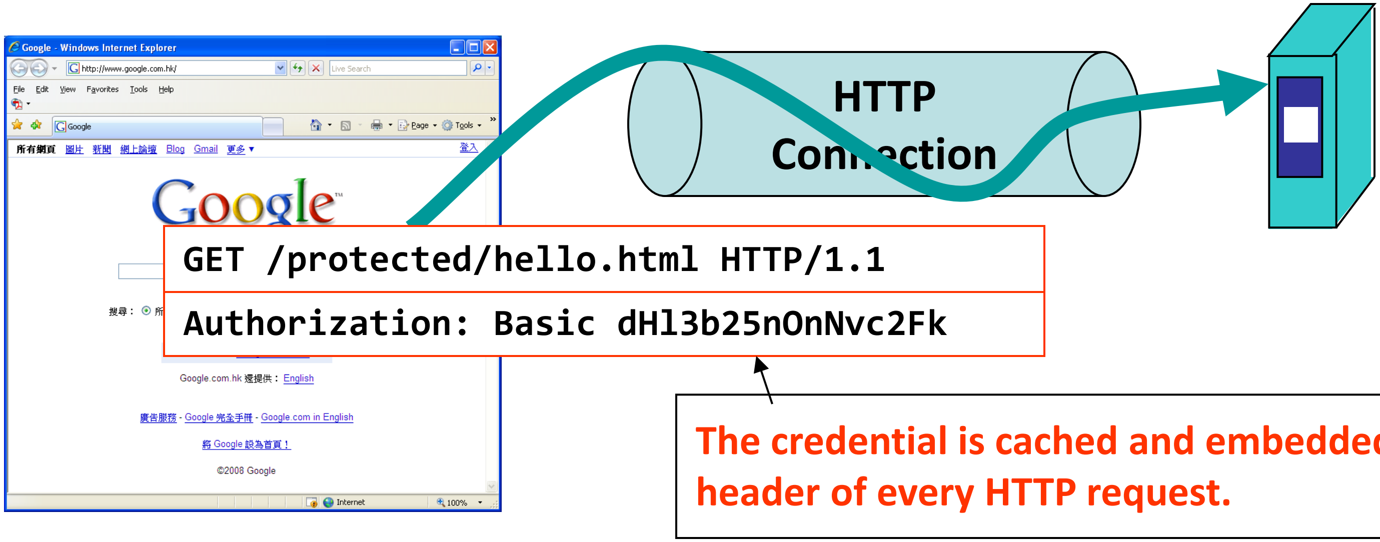


This finishes the initial login process…

After logged in…

- Let me ask… "do you need to log in when you access the protected page again?"

When the browser accesses a document in the same realm…
You don't need to log in again!


Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
Page Tools
所有網頁 圖片 新聞 網上論壇 Blog Gmail 更多 ▾
登入
Google
搜尋： ○ 所
Google.com.hk 還提供： English
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
©2008 Google
Internet
100%
HTTP
Connection

```
GET /protected/hello.html HTTP/1.1
Authorization: Basic dHl3b25nOnNvc2Fk
```

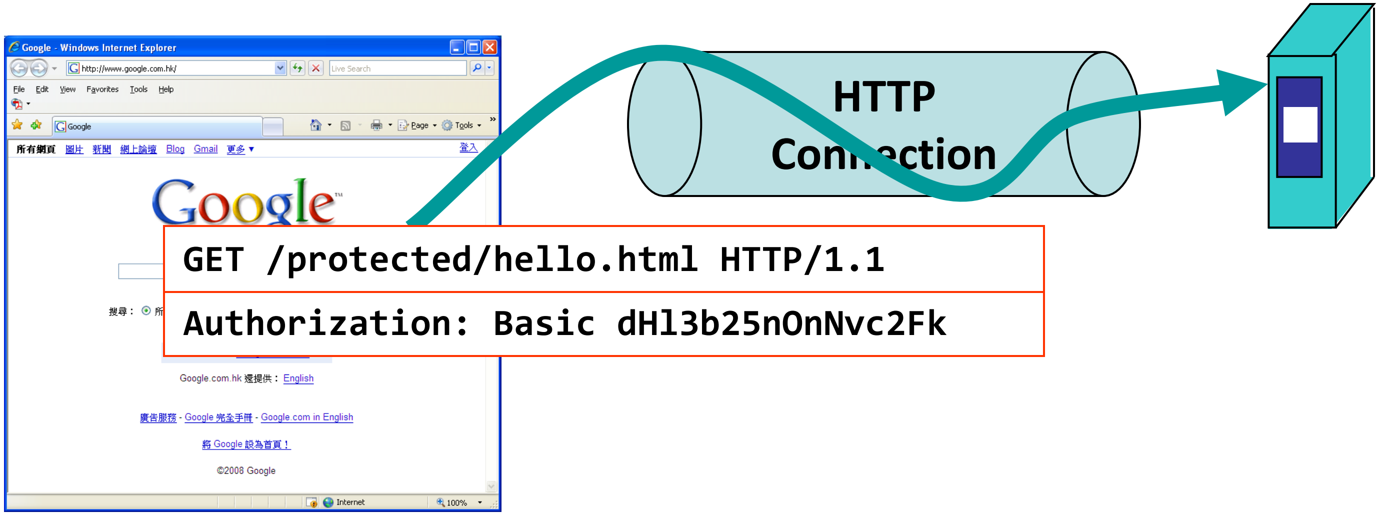
The credential is cached and embedded in the header of every HTTP request.

After logged in…

- What does “**`dH13b25nOnNvc2Fk`**” means?

```
[tywong@linux ] $ echo "dHl3b25nOnNvc2Fk" | base64 -d
tywong:sosad
```

The credential?!!



```
GET /protected/hello.html HTTP/1.1
Authorization: Basic dHl3b25nOnNvc2Fk
```

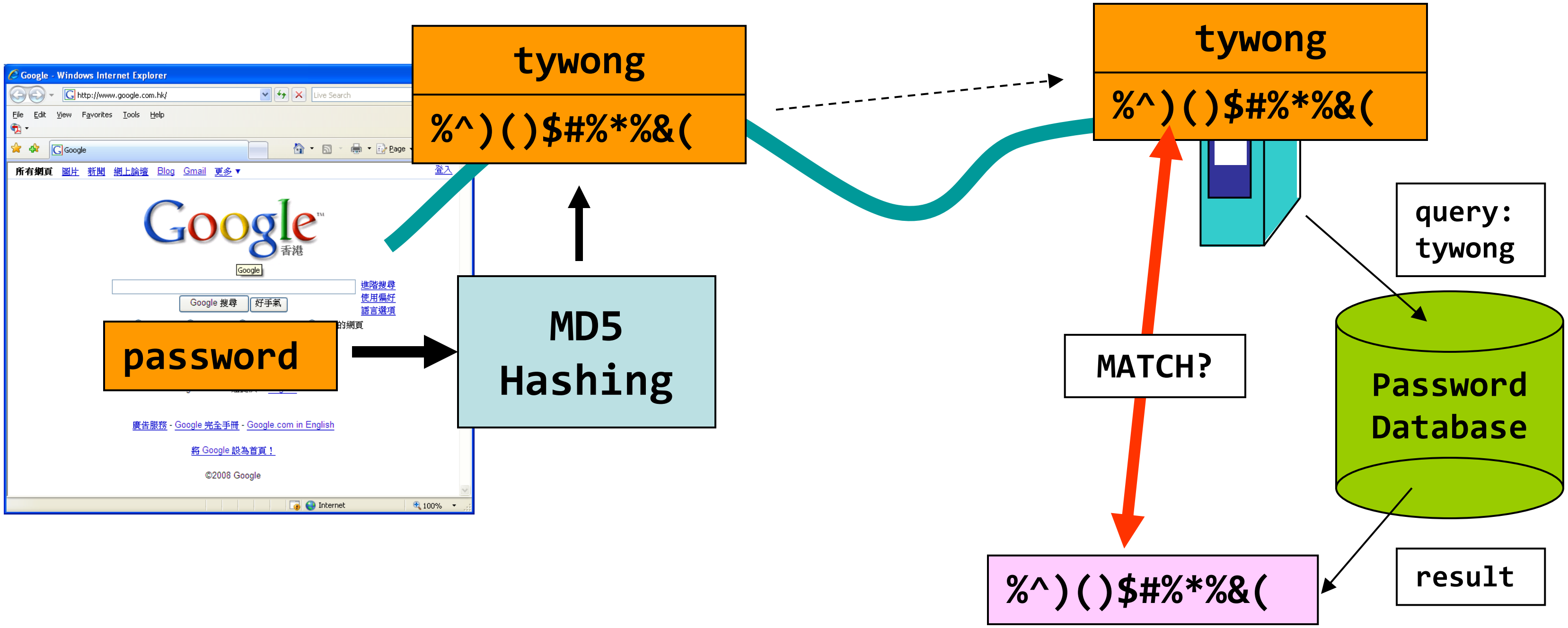
After logged in…

- What the HTTP protocol is doing?!
  - **It exposes the credential in the air!**
  - So, you should avoid using the “**`Basic`**” type but uses the “**`Digest`**” type HTTP Authentication.

- To make a long story short…
  - “**`Digest`**” uses **hashing**, instead of encoding.

“**Digest**” HTTP Authentication

- No plain-text password should be found in transit.



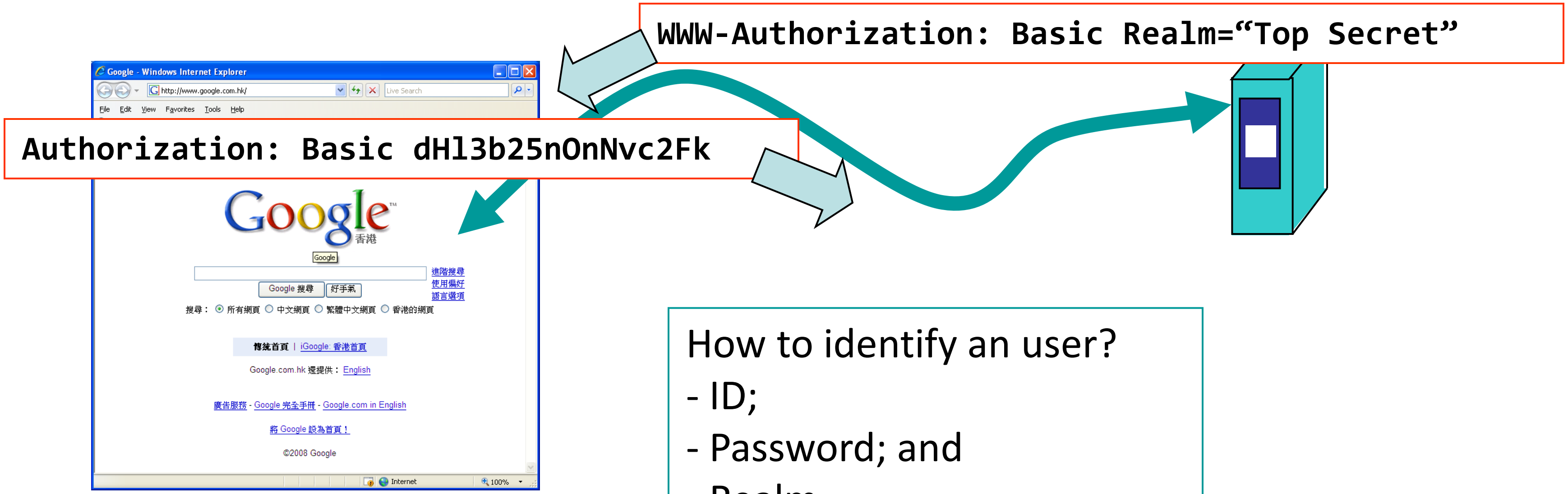
“**Digest**” HTTP Authentication

- We don’t plan to cover much about the “**Digest**” authentication because:
  - It needs an extra module for the web server, which is not always available.
  - It needs the program “**htdigest**” to produce the password file.
    - I can’t find this program on CSE dept UNIX workstations…but it is found in Linux workstations!
  - Nevertheless…you are recommended to use “**Digest**”!

[Example] “protected_digest/”

Summary on logging in…

- Methodology:
  - An on-demand type of authentication.



Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google 香港
Google 搜尋
好手氣
進階搜尋
使用偏好
語言選項
搜尋： ● 所有網頁 ○ 中文網頁 ○ 繁體中文網頁 ○ 香港的網頁
傳統首頁 | iGoogle: 香港首頁
Google.com.hk 還提供： English
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
©2008 Google
Internet
100%

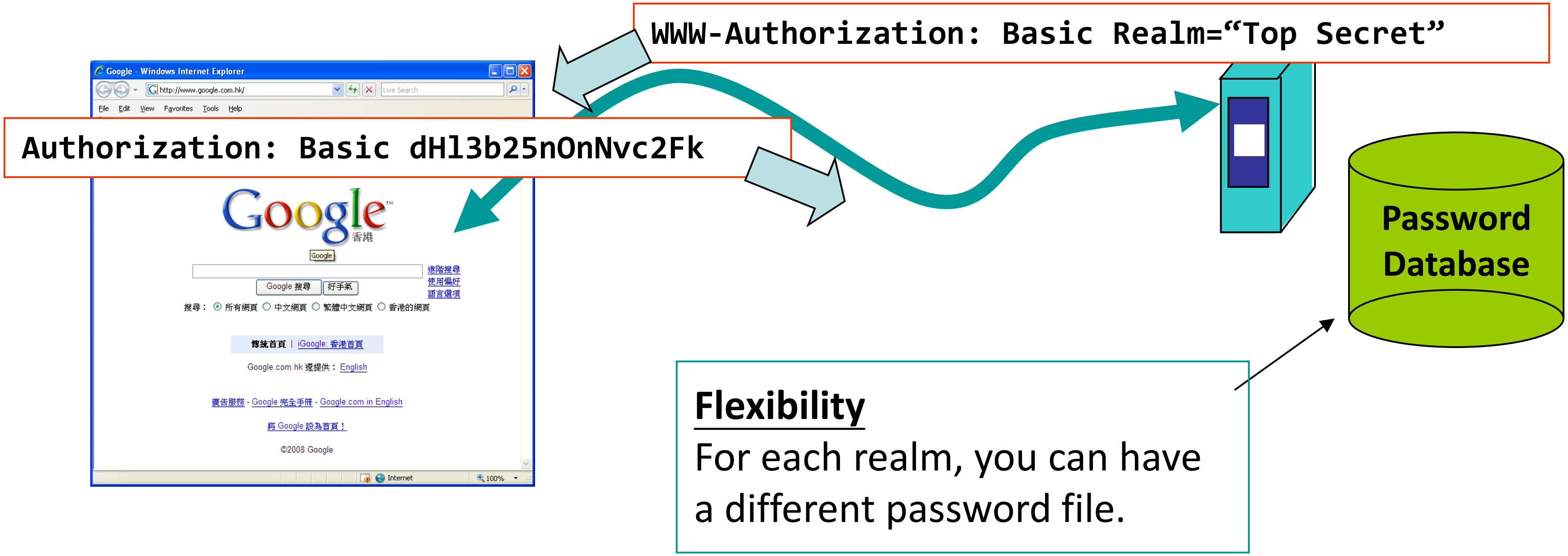
```
WWW-Authorization: Basic Realm="Top Secret"
```

```
Authorization: Basic dHl3b25nOnNvc2Fk
```

How to identify an user?
- ID;
- Password; and
- Realm.

Summary on logging in...

- Password Management:
  - Has tailor-made program to manage.



```
WWW-Authorization: Basic Realm="Top Secret"
```

```
Authorization: Basic dHl3b25nOnNvc2Fk
```

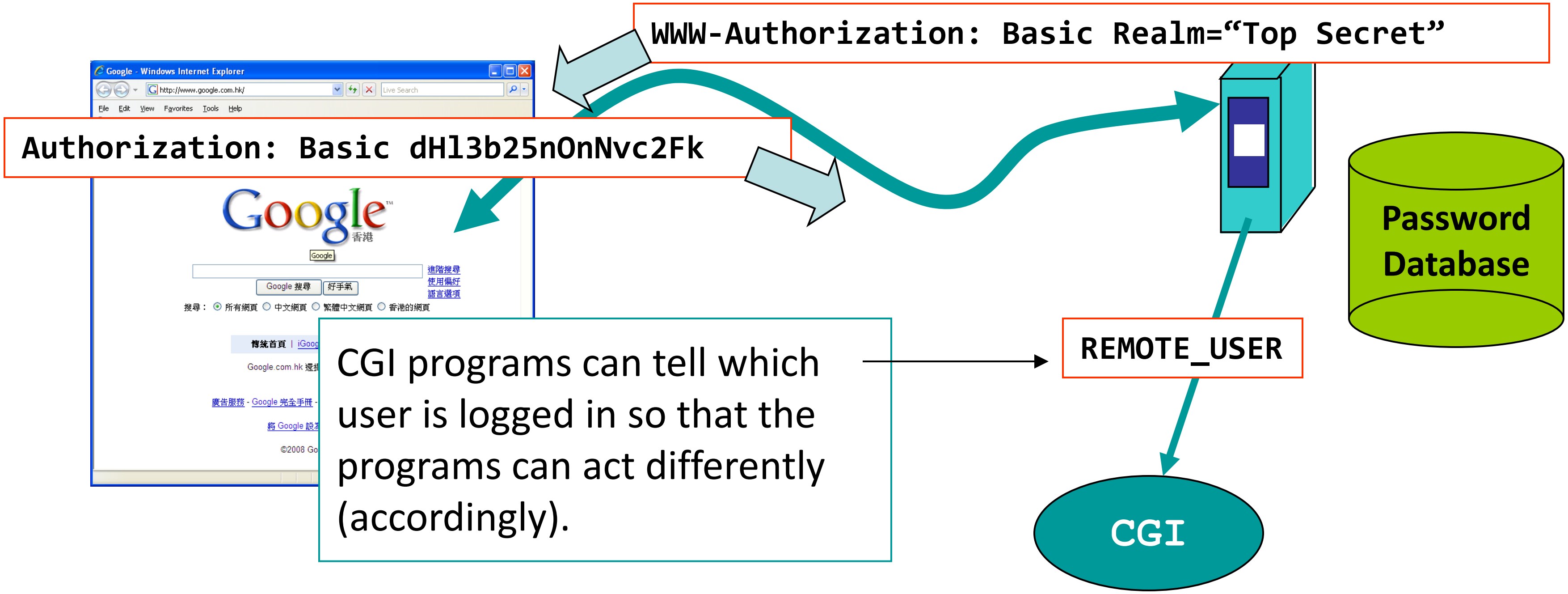
Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google 香港
Google
進階搜尋
使用偏好
語言選項
Google 搜尋
好手氣
搜尋： ● 所有網頁 ○ 中文網頁 ○ 繁體中文網頁 ○ 香港的網頁
傳統首頁 | iGoogle: 香港首頁
Google.com.hk 還提供： English
廣告服務 - Google 完全手冊 - Google.com in English
將 Google 設為首頁！
©2008 Google
Internet
100%
Password Database

Flexibility

For each realm, you can have a different password file.

Summary on logging in…

- Session Management:
  - CGI program can tell which user is logged in.



Google - Windows Internet Explorer
http://www.google.com.hk/
Live Search
File Edit View Favorites Tools Help
Google
香港
Google 搜尋
好手氣
進階搜尋
使用偏好
語言選項
搜尋： 所有網頁 中文網頁 繁體中文網頁 香港的網頁
Password Database
CGI

WWW-Authorization: Basic Realm=“Top Secret”

Authorization: Basic dHl3b25nOnNvc2Fk

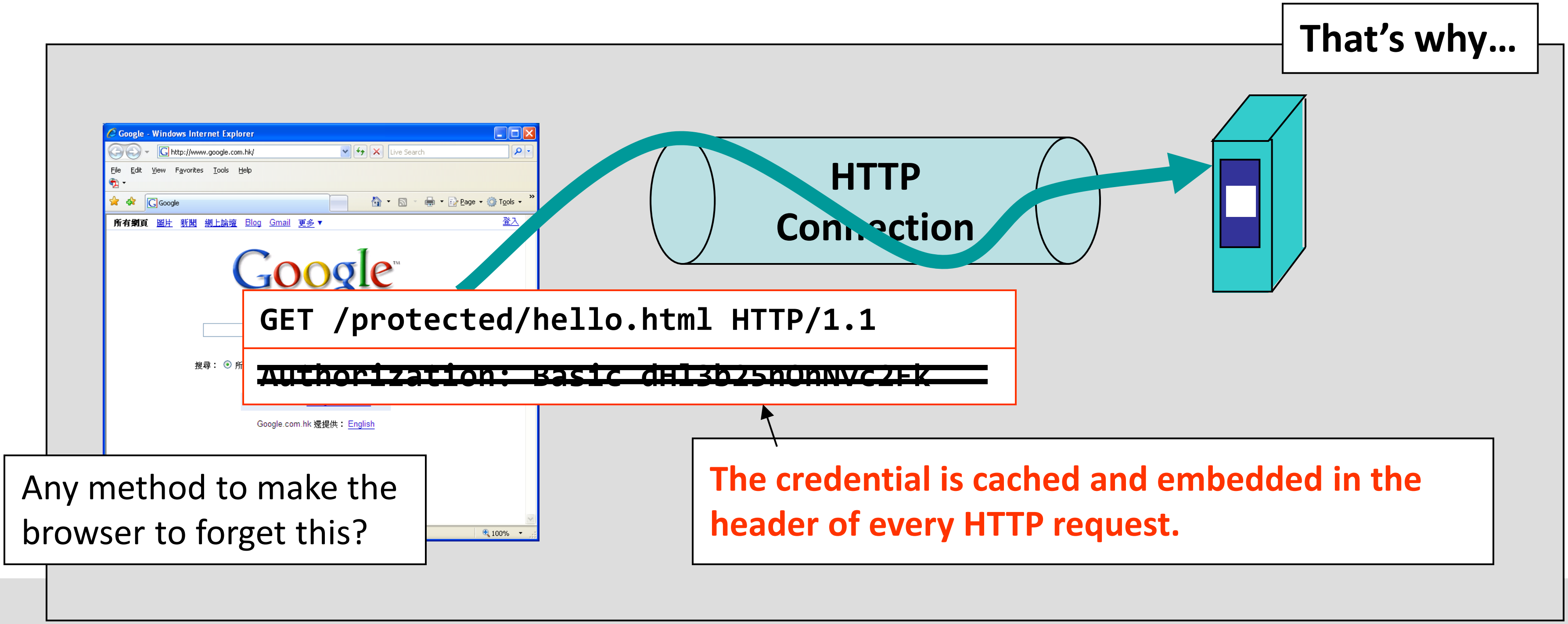
CGI programs can tell which user is logged in so that the programs can act differently (accordingly).

REMOTE_USER

Logging out…

- Easy, shut the browser down.
  - Hey, I said "**browser**", not a "**tab**"!

- But, why shutting down the browser?



That's why…

GET /protected/hello.html HTTP/1.1

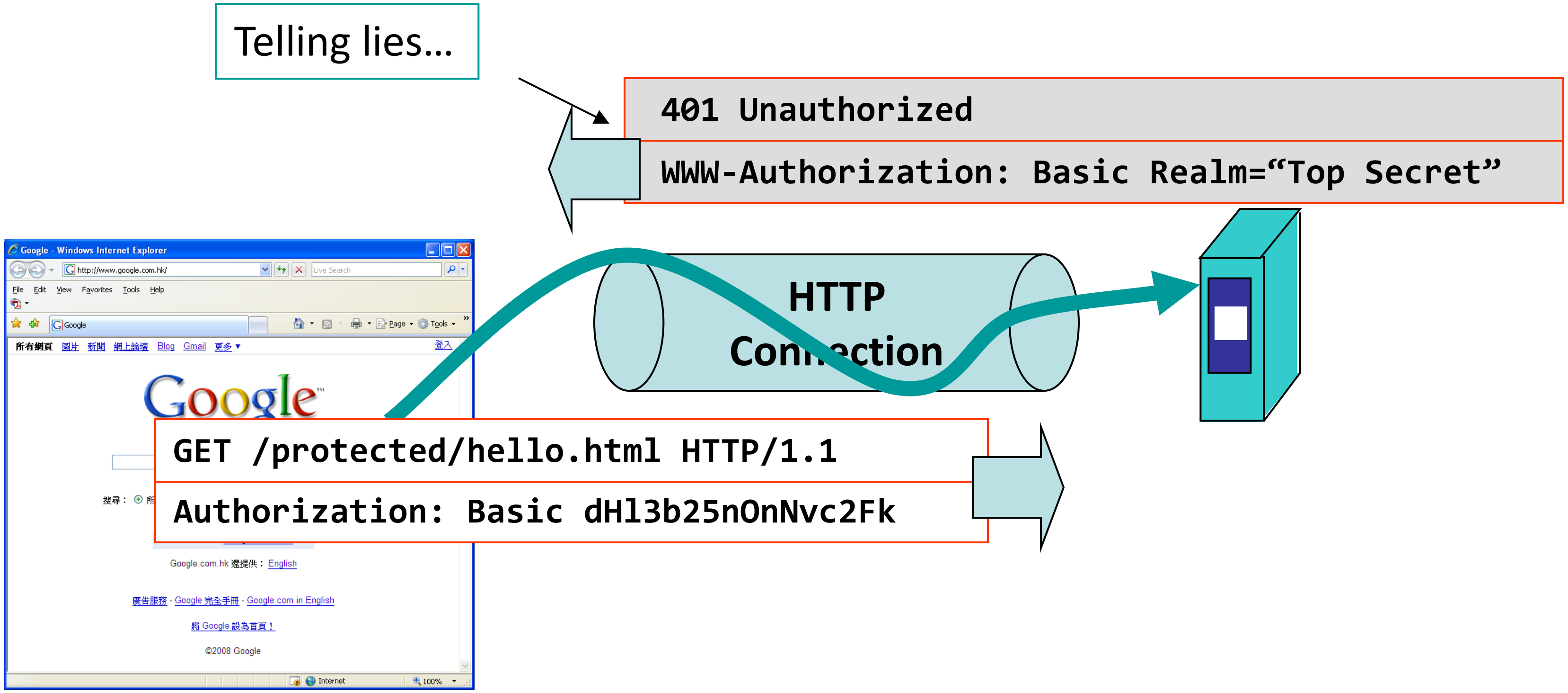
~~Authorization: Basic dHl3b25hOnNvc2Fk~~

The credential is cached and embedded in the header of every HTTP request.

Any method to make the browser to forget this?

Logging out…

- Rather easy…if the server can tell lies…
  - But, how…



Telling lies…

```
401 Unauthorized
WWW-Authorization: Basic Realm=“Top Secret”
```

```
GET /protected/hello.html HTTP/1.1
Authorization: Basic dHl3b25nOnNvc2Fk
```

Logging out…

- Rather easy…if the server can tell lies…
  - But, how…
  - Using CGI…

Sending this out will cheat the browser. The browser thinks that the previous credential is wrong and ceases to cache it.

```
$realm = "Top Secret";

print "Status: 401 Unauthorized\n";
print "WWW-Authenticate: Basic realm=\"${realm}\"\n";
print "Content-type: text/html\n\n";

print ......
```

[Example] “protected_basic/logout.cgi”

- References:
  - Apache documentation

http://httpd.apache.org/docs/2.0/howto/auth.html

Bonus track: user tracking....

- References:

http://w2.eff.org/Privacy/Marketing/web_bug.html

http://code.google.com/apis/analytics/docs/concepts/gaConceptsOverview.html

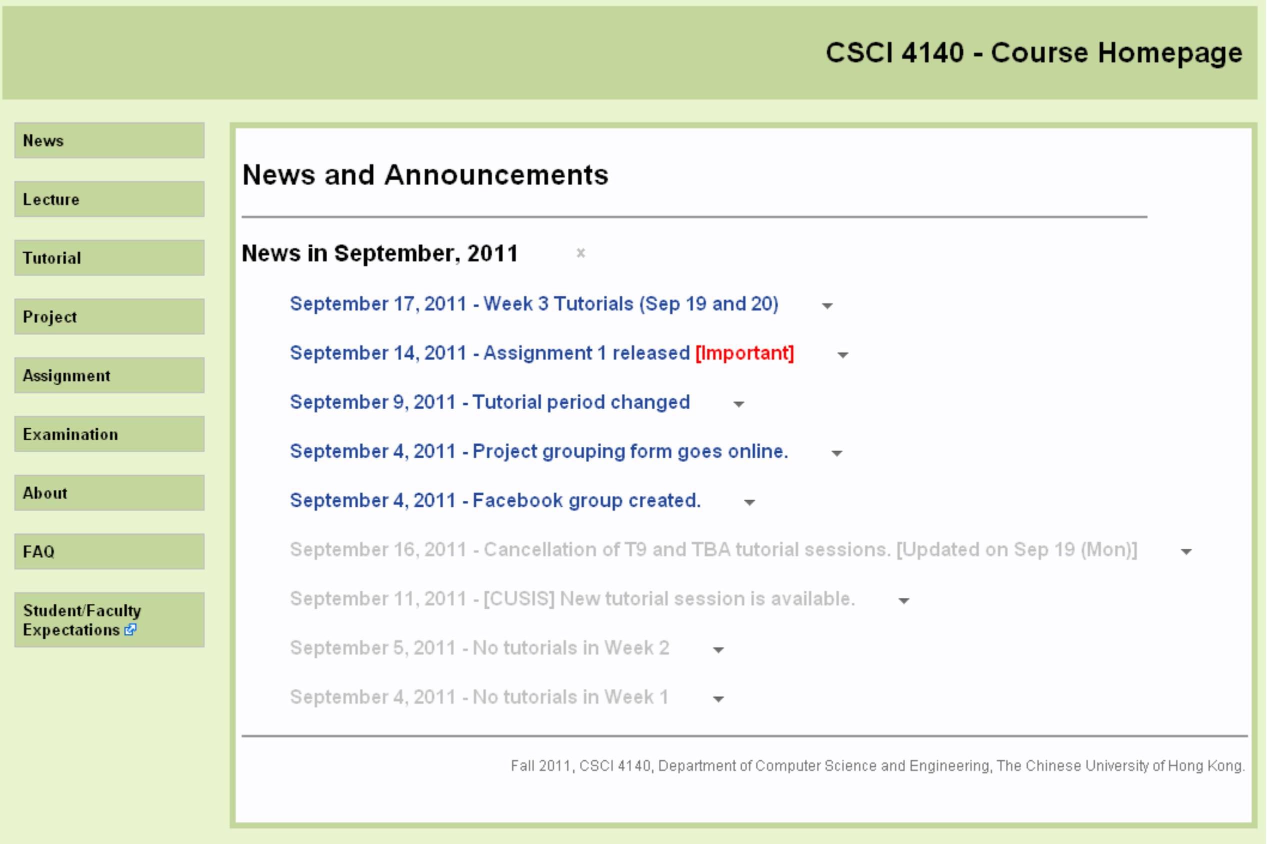
I don't plan to release any codes since this is a controversial topic. But, there could be some small examples...

Tracking Cookie

- the evil and common use of HTTP cookies.

User tracking ... why?

E.g., Owner of this page wants to find out how many visits on the assignment specification...



CSCI 4140 - Course Homepage

News and Announcements

News in September, 2011

September 17, 2011 - Week 3 Tutorials (Sep 19 and 20)

September 14, 2011 - Assignment 1 released [Important]

September 9, 2011 - Tutorial period changed

September 4, 2011 - Project grouping form goes online.

September 4, 2011 - Facebook group created.

September 16, 2011 - Cancellation of T9 and TBA tutorial sessions. [Updated on Sep 19 (Mon)]

September 11, 2011 - [CUSIS] New tutorial session is available.

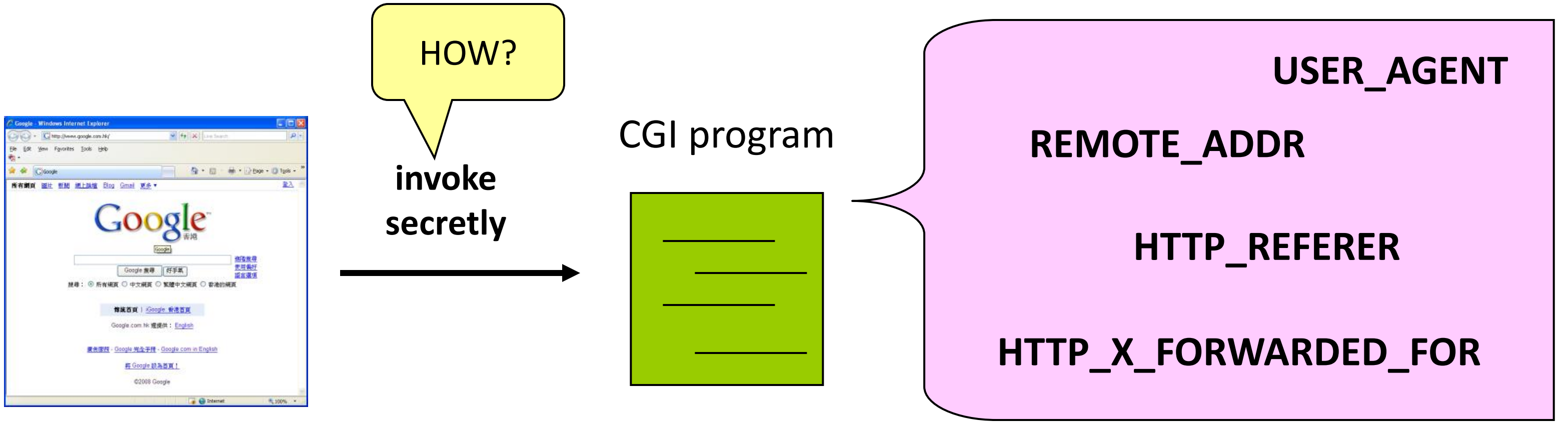
September 5, 2011 - No tutorials in Week 2

September 4, 2011 - No tutorials in Week 1

Fall 2011, CSCI 4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

User tracking … why?

Environment variable	Tracking purpose
USER_AGENT	The browser type & OS type.
REMOTE_ADDR	The source / proxy IP address.
HTTP_REFERER	The page that invokes this script.
HTTP_X_FORWARDED_FOR	The IP address that the proxy server is serving, if any.


HOW?

invoke secretly

CGI program

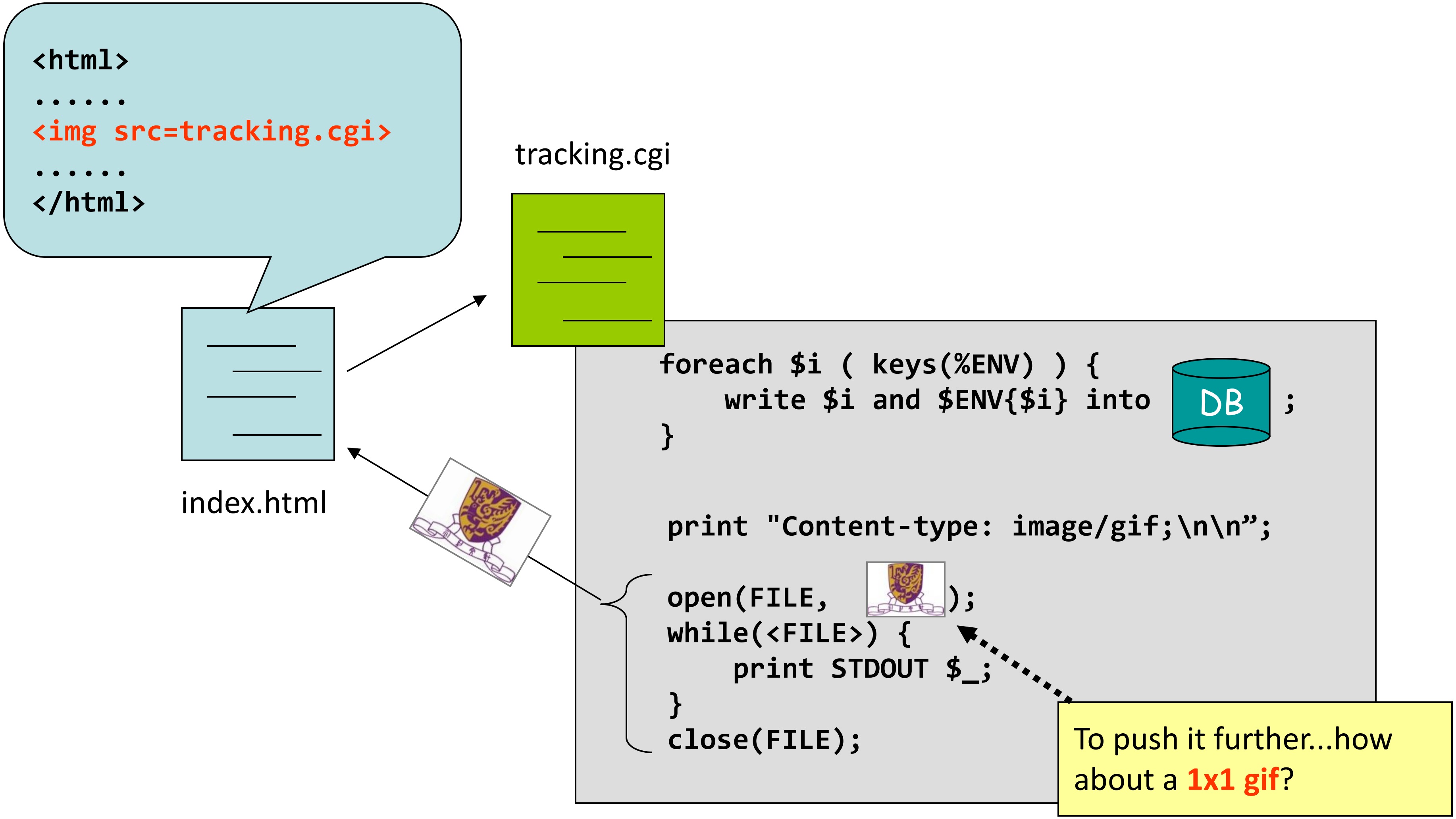
USER_AGENT

REMOTE_ADDR

HTTP_REFERER

HTTP_X_FORWARDED_FOR

User tracking … how?



```
<html>
......
<img src=tracking.cgi>
......
</html>
```

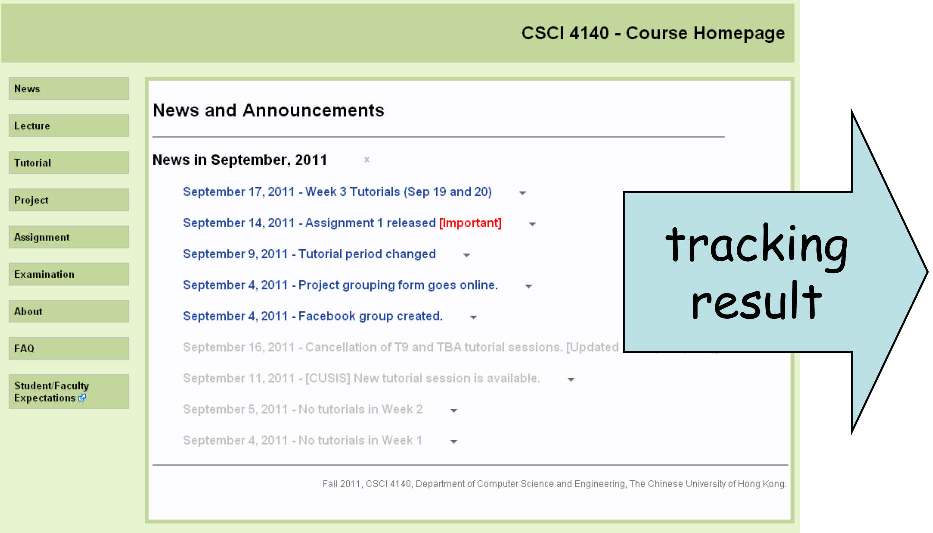
```
foreach $i ( keys(%ENV) ) {
    write $i and $ENV{$i} into DB ;
}

print "Content-type: image/gif;\n\n”;

open(FILE,      );
while(<FILE>) {
    print STDOUT $_;
}
close(FILE);
```

To push it further…how about a **1x1 gif**?

User tracking … so?

- The visit counts can never entertain the owner’s peeking desire…

- How about…tracking the **browsing behavior/habit of a particular user** on that course homepage?



CSCI 4140 - Course Homepage

News
Lecture
Tutorial
Project
Assignment
Examination
About
FAQ
Student/Faculty Expectations

News and Announcements

News in September, 2011

September 17, 2011 - Week 3 Tutorials (Sep 19 and 20)
September 14, 2011 - Assignment 1 released [Important]
September 9, 2011 - Tutorial period changed
September 4, 2011 - Project grouping form goes online.
September 4, 2011 - Facebook group created.
September 16, 2011 - Cancellation of T9 and TBA tutorial sessions. [Updated
September 11, 2011 - [CUSIS] New tutorial session is available.
September 5, 2011 - No tutorials in Week 2
September 4, 2011 - No tutorials in Week 1

Fall 2011, CSCI 4140, Department of Computer Science and Engineering, The Chinese University of Hong Kong.

Content	Count
about.html	76 (5.01%)
assignment.html	175 (11.54%)
index.html	457 (30.15%)
……	……

User tracking … the final version…

```
$_ = $ENV{HTTP_COOKIE};
if(/tracking_id/) {
    create  tracking_id = ^$^(*()*&$  ;

    write  tracking_id = ^$^(*()*&$  into  DB  ;

}
else {
    update visit record of  tracking_id  into  DB  ;
}
```

tracking.cgi

```
foreach $i ( keys(%ENV) ) {
    write $i and $ENV{$i} into  DB  using  tracking_id as the key;
}


print “Set-Cookie: tracking_id: value; expires: 10 years later”;

print "Content-type: image/gif;\n\n”;
open(FILE,  );
......
```

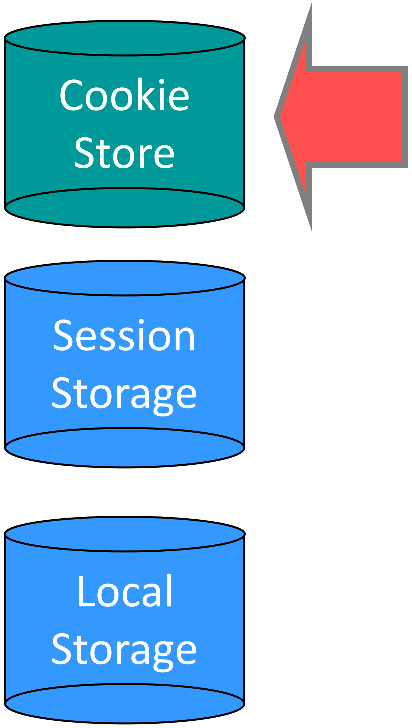
User tracking ... wow....

1st version	2nd version
Identify visitors by IP addresses. What if the machine changes its IP address?	**Identify visitors by cookies.** The tracking cookie will not change after the machine changes its IP. Usually, a person is loyal to a particular browser. This makes the tracking result more accurate...
	One can calculate how a visitor spends his/her time over a series of web pages...
	Can you think of / implement more?

Some Discussions

- Can anyone track who you are?
- What if the same tracking “gif” (a.k.a. a web bug) is deployed in all web sites?
- What if a web bug is deployed in emails using HTML format?
- Is it legal to do so?
- Last but not least, are tracking cookies harmful?

Future: HTML5 – **WebStorage**



Cookie

- Cookie is sent with every HTTP request.

- Cookie is set with every HTTP response.

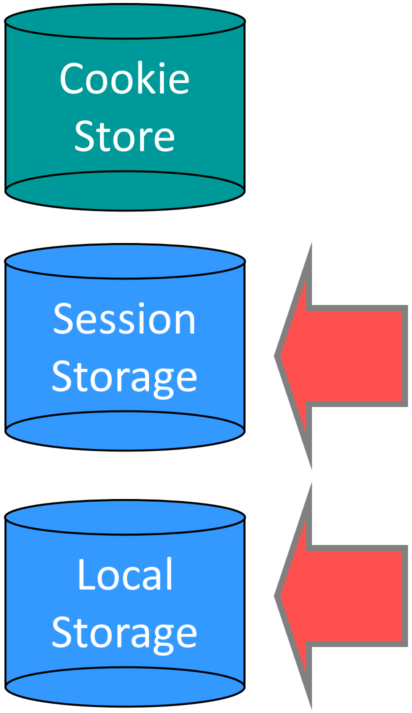
- But, HTTP header has a size limit.
E.g., Apache default limit is 8190 bytes.
(**`LimitRequestFieldSize`**)

- It is a global storage to the entire browser.

- It will expire.

http://dev.w3.org/html5/webstorage/

Future: HTML5 – **WebStorage**



sessionStorage & localStorage

- It is sent and stored using JavaScript only, not because of the HTTP header.

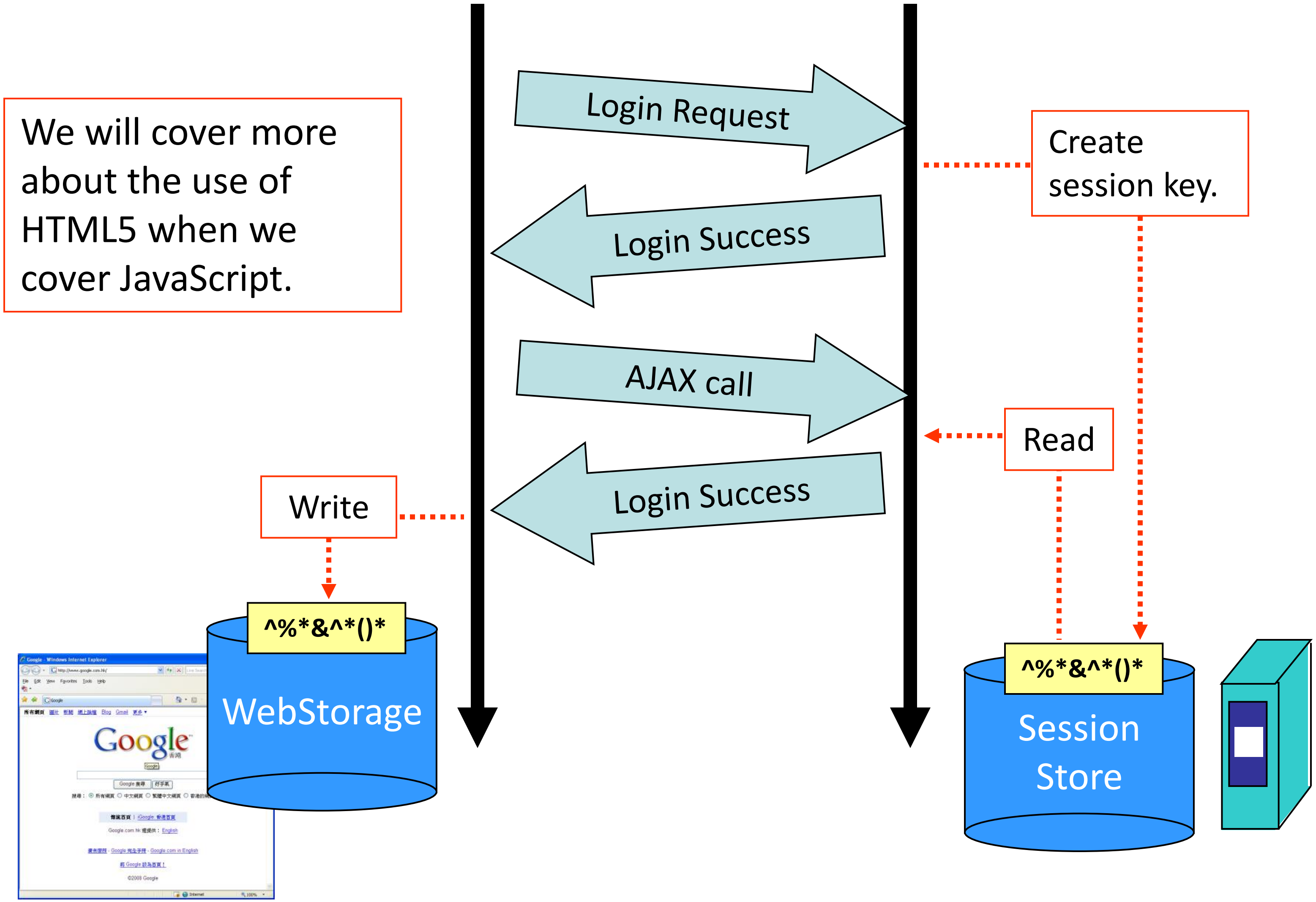
- sessionStorage is temporary; it expires when the tab or the browser is closed.

- localStorage is permanent; it can only remove using JavaScript!

http://dev.w3.org/html5/webstorage/

Future: HTML5 – **WebStorage**

We will cover more about the use of HTML5 when we cover JavaScript.



Future Readings...

- evercookie
  - http://samy.pl/evercookie/
- The Definitive Guide?
  - http://stackoverflow.com/questions/549/the-definitive-guide-to-forms-based-website-authentication