

Identity on the Web

"Il mio nome è Remo Williams"

"Alas. No cigarette ash, no fingerprints. Not a woman's glove, nor a vague one trail of perfume ... None of those things that novice detectives always find"

Hercule Poirot



Why trace people?

- ◆ Business optimization:
 - ◆ User profiling can aim to propose the "right" ads to a potentially buyer.
- ◆ Site optimization:
 - ◆ User profiling and surf history can aim to optimize the structure of the site; for example, if a class of users visit often and almost always a very deep zone of your site, you should consider to refactor the site in order to move up that zone.
- ◆ Surf optimization:
 - ◆ User profiling permits to personalize the site presentation.

Trace without help



How trace people?

How Web Surfing works (a fly-by):

- ◆ Page (HTML) fly over HTTP(S)
- ◆ HTTP(S) is stateless (more or less) protocol
 - ◆ client shots (many times) and after serve it, the server forget all.
- ◆ HTTP(S) fly over TCP protocol
 - ◆ TCP is a stateful protocol: this guarantees that pieces of the page are reassembled on the client in a correct way
- ◆ TCP fly thanks to IP addresses (and TCP ports)

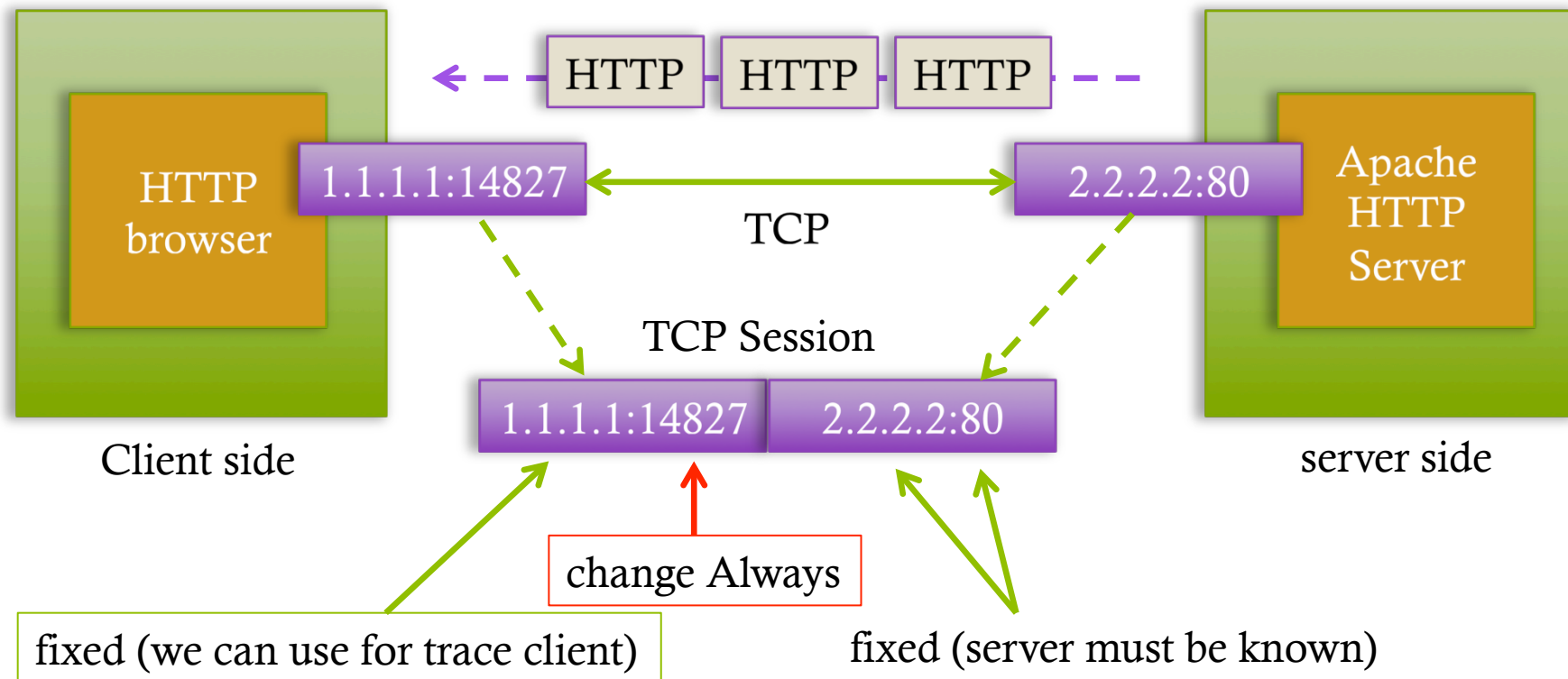
client IP and its TCP ports can be traced naively (if you are lucky...)

Trace connection

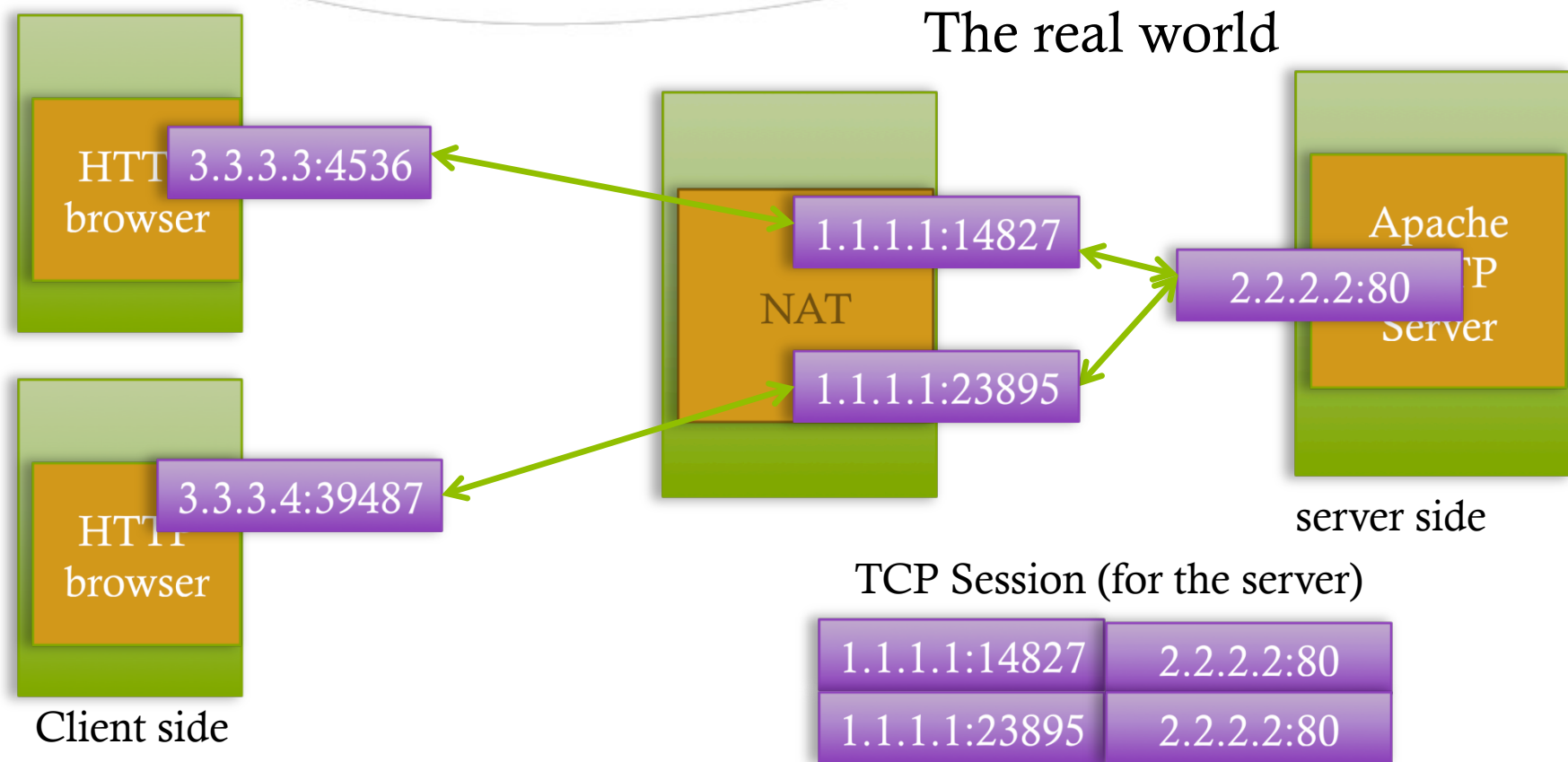
- ◆ We can (try to) trace client IP and port
 - ◆ IP gives some info on client geographical position
 - ◆ Usually IP changes quickly, even more quickly on mobile
 - ◆ May be IP and port is not owned by the client but by a node in the middle that act as proxy/VPN server
 - ◆ May be IP and port are shared among many clients when there is a Proxy or a PAT server

Transmit a HTTP Page

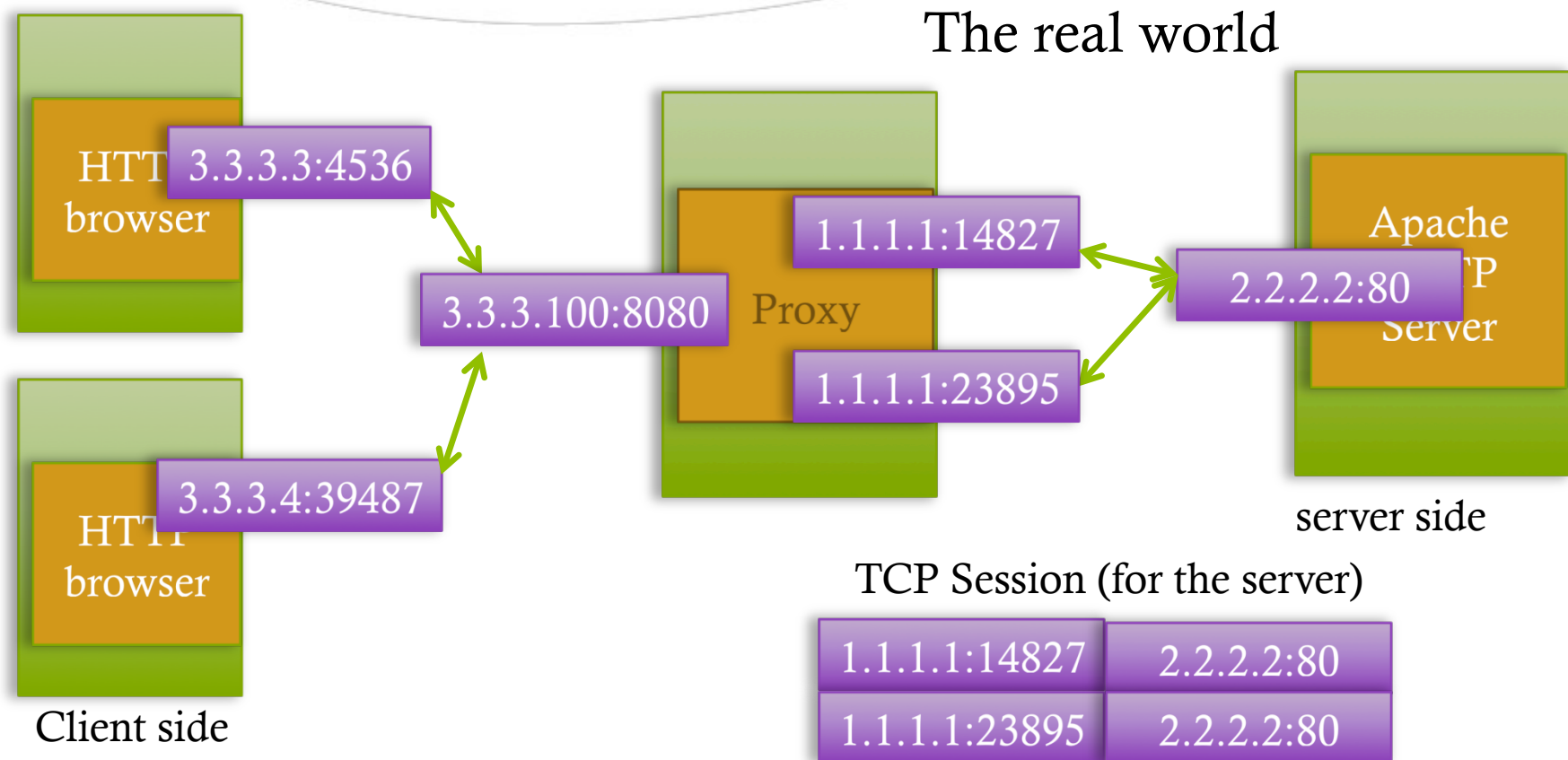
This beautiful world (in our dreams)



Client behind a NAT/VPN



Client behind a Proxy



Trace browser

- ◆ When browser present itself to the server, it give some informations about its capability, by the header field User-Agent:

User-Agent: <product> / <product-version> <comment>

Common format for web browsers:

User-Agent: Mozilla/<version> (<system-information>)
<platform> (<platform-details>) <extensions>

Examples:

Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0

Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101 Firefox/42.0

- ◆ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>
- ◆ https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- ◆ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent>

Trace with help



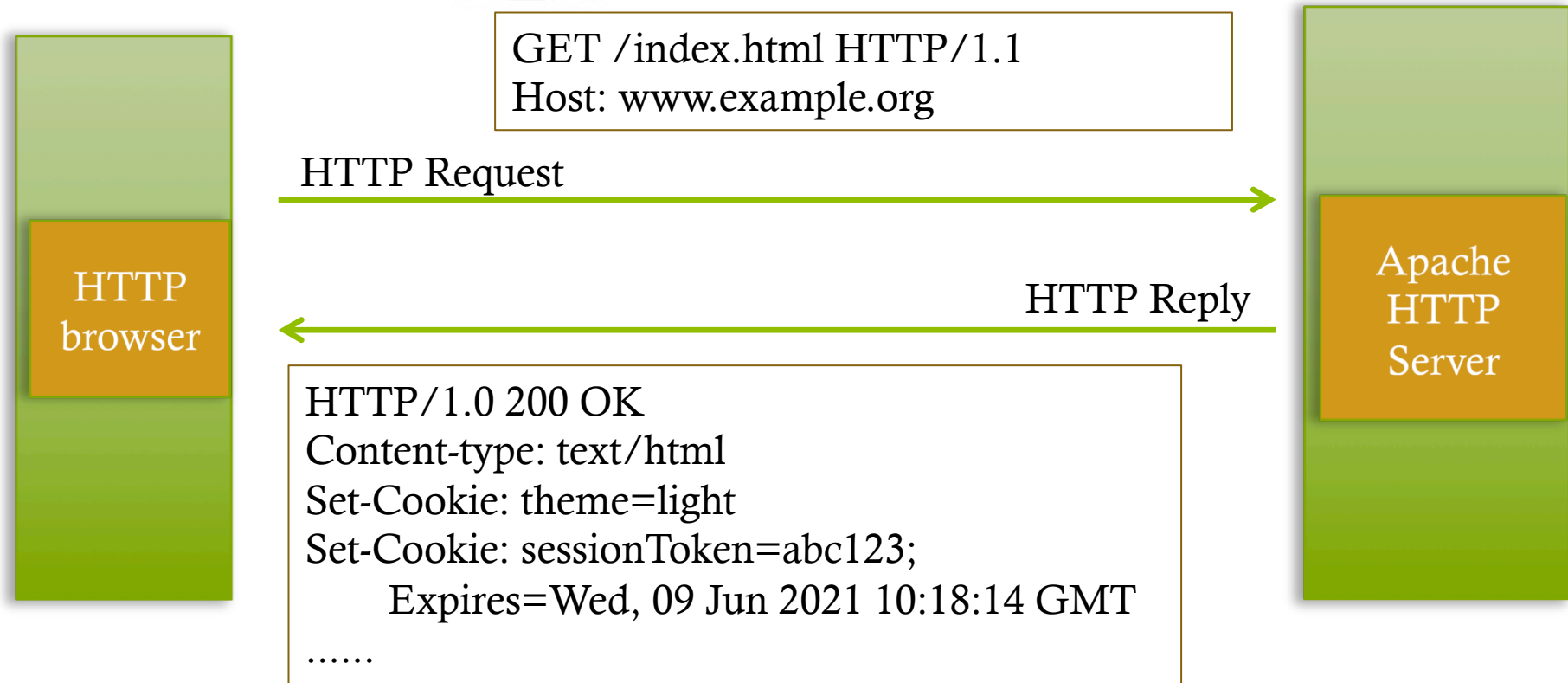
Cookies

- ◆ Trace people without help is unreliable
- ◆ A cookie is a small piece of text, sended by server and stored on the user's computer by their browser.

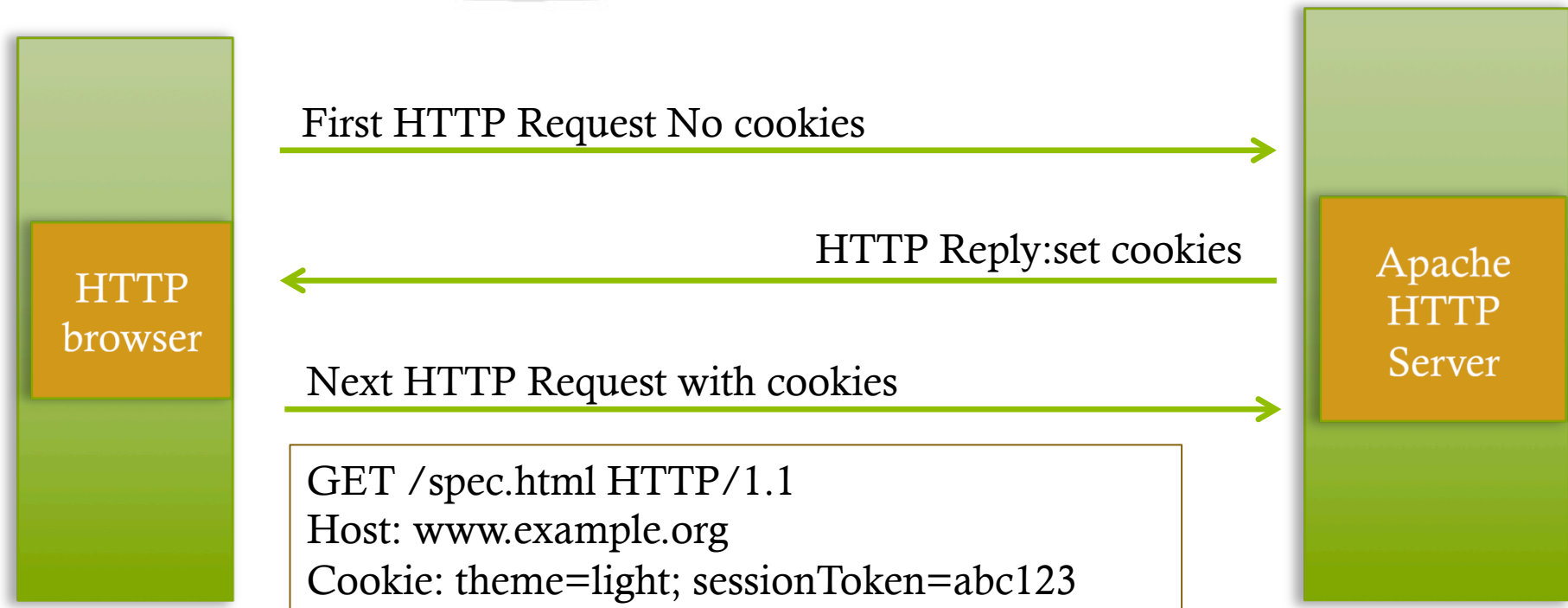
```
theme=light; sessionId=abc123
```

- ◆ When client ask something to a server, cookies from that server are added to the header of the HTTP request.

Setting Cookies



Using Cookies



Cookie Types

- ◆ **Session cookie:** cookie die after close browser
 - ◆ **Persistent cookie:** cookie die after fixed time
 - ◆ **Secure cookie:** cookie is sent only on HTTPS
 - ◆ **HttpOnly cookie:** cookie is not accessible to script
 - ◆ **SameSite cookie:** cookie is sent only to originate site
 - ◆ **Third-party cookie:** server send cookies for other sites
 - ◆ **Supercookie:** cookie for top domains, such as .com
 - ◆ **Zombie cookie:** cookie that if deleted, then it is recreated
- ◆ https://en.wikipedia.org/wiki/HTTP_cookie

Cookies vs Identity

- ◆ Cookie permits to server to recognize a particular instance of browser in a particular device during time; without other actions, the user identity is anonymous.
- ◆ If the client authenticates itself by some login process and the server associates this authentication to the understanding cookie, then the cookie became the proof of the user's identity and the protection of this *powerful* cookie became important.
- ◆ The **Secure**, **HttpOnly** and **SameSite** cookie types address various security concerns about malicious site which try to steal or misuse cookies.

EU cookie directive

- ◆ In 2002, the European Union launched the Directive on Privacy and Electronic Communications, a policy requiring end users' consent for the placement of cookies, and similar technologies for storing and accessing information on users' equipment.
- ◆ In particular, Article 5 Paragraph 3 mandates that storing data in a user's computer can only be done if the user is provided information about how this data is used, and the user is given the possibility of denying this storing operation.

Trust User Identity

"I am who I claim to be"



An identity problem

- ◆ Surfing on the web does not require particular knowledge about identities about clients and servers.
- ◆ Identity became a problem when some resources can be managed only by authorized users:
 - ◆ moving money from a bank account to another
 - ◆ watching private photos
 - ◆ reading news sites accessible only by subscription.

How can trust a site who claims to be the portal of my bank?
How my bank trust myself during my account operations?

Simmetric Key

- ◆ A symmetric key is a secret code known by both of the two sides of a communication channel.
- ◆ In examples, the two sides of the channel are historically called "Alice" and "Bob"
 - ◆ Sometimes it needs to introduce a third actor, "Carl", often with the role of a robber/hacker/bad guy.
- ◆ Alice wants to tell something really important and secret to Bob through this communication channel, but Bob wants to be sure that Alice is really Alice before accepting the message.
 - ◆ Carl usually wants to know or falsify the secret message, impersonates Alice and does other such bad things.

Simmetric Key

Alice

"I'm Alice"

My secret key is "fuffi"

Bob

"Give me the proof!"

It match with ones I know ("fuffi")!,
Welcome, Alice!

Simmetric Key

- ◆ Some security concerns:
 - ◆ How securely share the secret shared key between Alice and Bob?
 - ◆ What's happens if someone (Carl) tap the key?
 - ◆ What's happens if Bob misuse the key?
- ◆ Some advantage:
 - ◆ Easy to implements
 - ◆ Cheap in term of cpu power consumption.

Carl steal secret key

Alice wants to speak with Bob on an insecure (HTTP) channel.

Alice

Carl

Bob

"I'm Alice"

(...interesting...)

"Give me the proof!"

My secret key is "fuffi"

(I tap "fuffi"!!! Now I can became Alice!!!)

It match with ones I know ("fuffi")!,
Welcome, Alice!

Carl impersonates Alice

Alice wants to speak with Bob on an insecure (HTTP) channel.

Carl

Bob

"I'm Alice"

(you cannot recognise me, heheheh)

"Give me the proof!"

My secret key is "fuffi"

(I tap the secret key!!! poor Alice!!)

It match with ones I know ("fuffi")!,
Welcome, Alice!

Even Bob impersonates Alice

Dan is the Alice's banker, Bob want to steal money to Alice

Bob

"I'm Alice"

(you cannot recognise me, heheheh)

My secret key is "fuffi"

(I dont need to tap the secret key, I know it!!! poor Alice!!)

I want to retrieve all my money

(Alice's' money of course, heheheh)

Dan

"Give me the proof!"

It match with ones I know ("fuffi")!,

Welcome, Alice!"

Asymmetric Key

- ◆ To address the concerns on using symmetric key, we can use asymmetric keys.
- ◆ The asymmetric key is composed by two part:
 - ◆ The private key known only by Alice,
 - ◆ The public key public sharable.
- ◆ With the asymmetric key, It is possible to sign *a token* with one part and verifying the token sign with the other.

Asymmetric Key

Alice wants to speak with Bob on an insecure (HTTP) channel.
The private key of Alice is "3", the public key is "6",

Alice

"I'm Alice"

mumble... $\text{pow}(\text{pow}(5,3),2)=15625$

"I sign the stone '5' and the result '15625'"

Bob

"Give me the proof!"

mumble... $\text{radix}(15625,6)=5$

It matches with "5"!,

Welcome, Alice!"

Certification Authority

"Who's on the first base? Who's on the first base!"

Alice on the air "I sign this stone with my (secret) sign"

Bob: "What is your (public) sign?"

Alice: "This is my (public) key! It's signed by (secret) sign of Carl"

Bob: "Ok, but what is the (public) sign of Carl?"

Alice: "This is the (public) key of Carl! It's signed by (secret) sign of Dan?"

Bob: "Oh Ok, I know Dan! good!"

- ◆ We need of a Trusted Certification Authority (CA) (*Dan*)

Server Identity

- ◆ When a client connect a server, it may want to grant its identity
- ◆ SSL/TLS is a protocol to grants Identity by trust (a set of) CA, who is responsible to verify the identity claimed by a Server

Server: I give you my public key signed by CA

Client: I trust CA, ok.

Client: I cript (sign) "a strong secret word" with your public key, what is the secret word?

Server: I decrypt your secret word with my secret key, Its "a strong secret word"

Client: Ok, You are who you claim!

Server: And You??? Who you are?

Client Identity

- ◆ How trust client identity:
 - ◆ SSL certificate, as server..... too complicate (and expensive)
 - ◆ A couple <user,password>, more better but if someone steal password?
 - ◆ two factor authentication: I authenticate the user twice, first by user password and after by some other shared secret:
 - ◆ phisical token
 - ◆ two way chanel communication
 - ◆ biometrical password (fingerprint, voice scan, face scan, sign/ gesture scan)