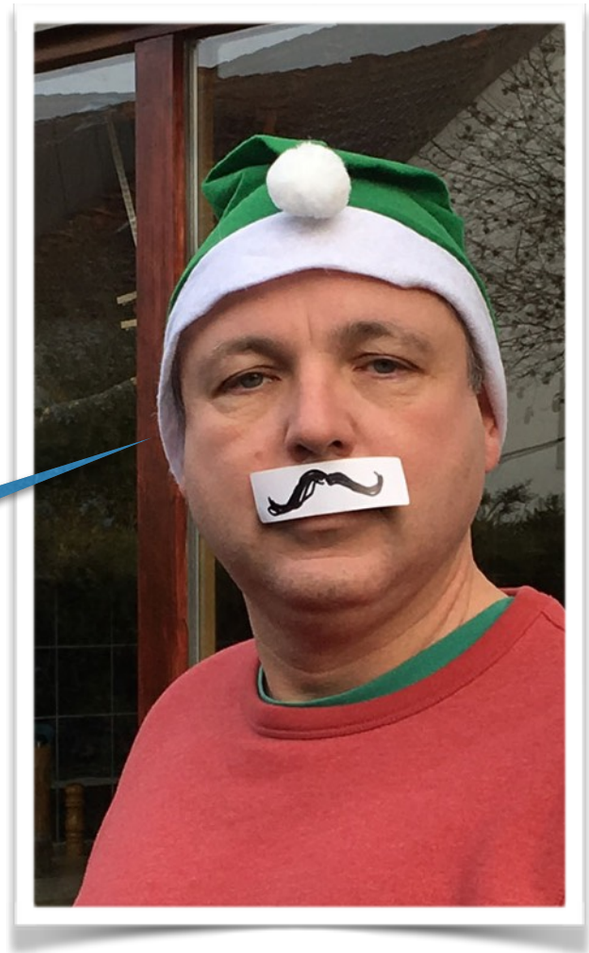
Big numbers as safety belt in cookies

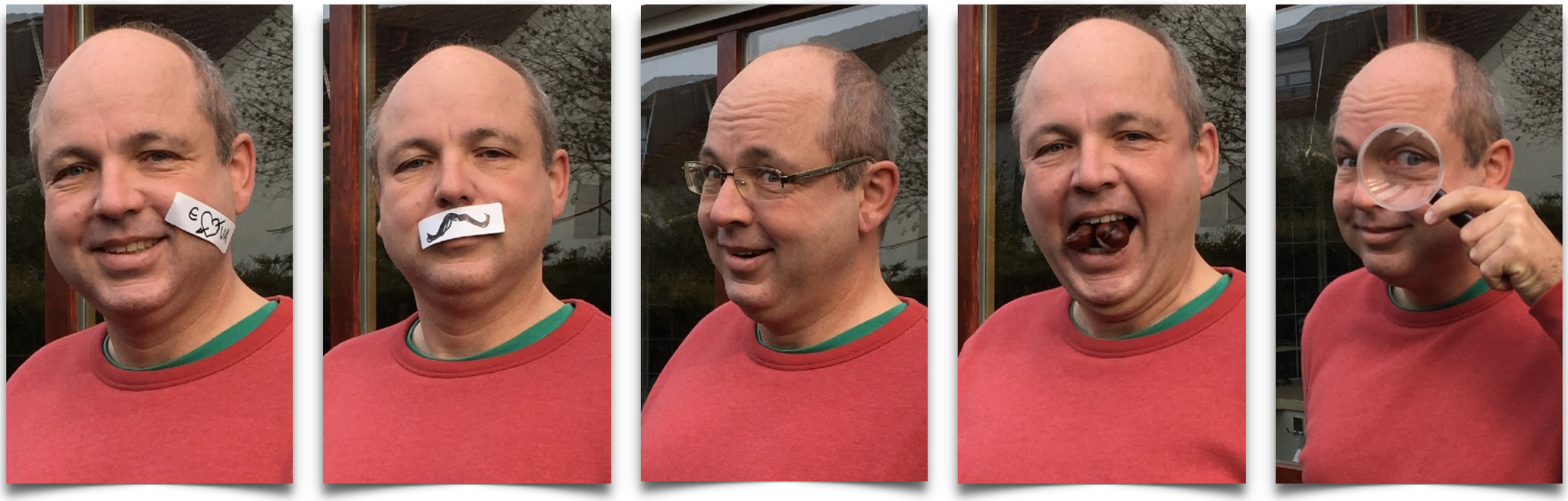
Richard Lippmann
richard.lippmann at posteo.de

for
KNF-Kongress, Nuernberg, Germany 11/2016

There is something wrong when you see a lot of dwarfs with green caps!

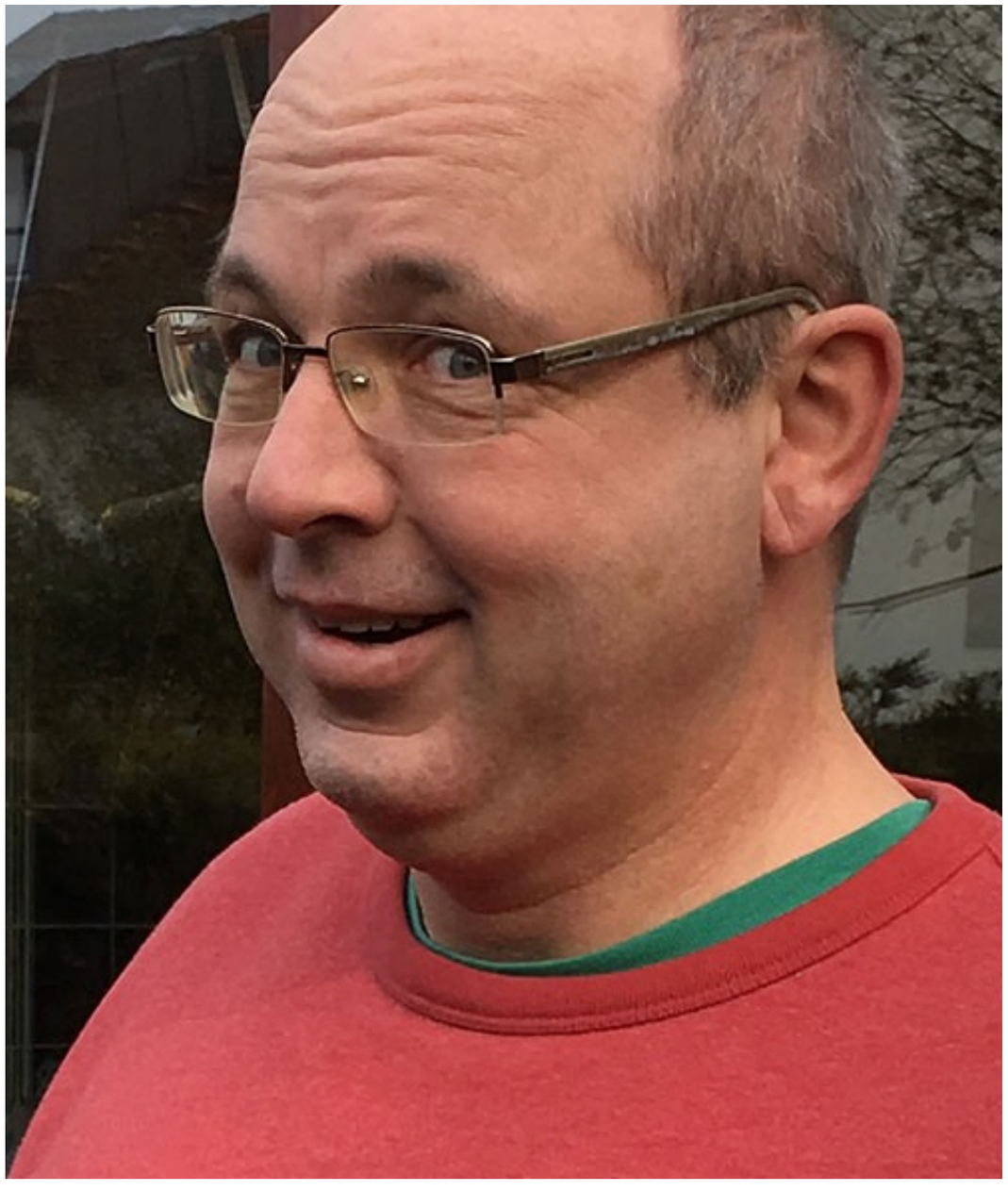


How does a server in a web-environment knows a user?



Cookies in web applications

- A server is talking to a lot of different users
- A user should be recognized in a secure way
- Also when being mobile
- Also when being behind a proxyserver
- …



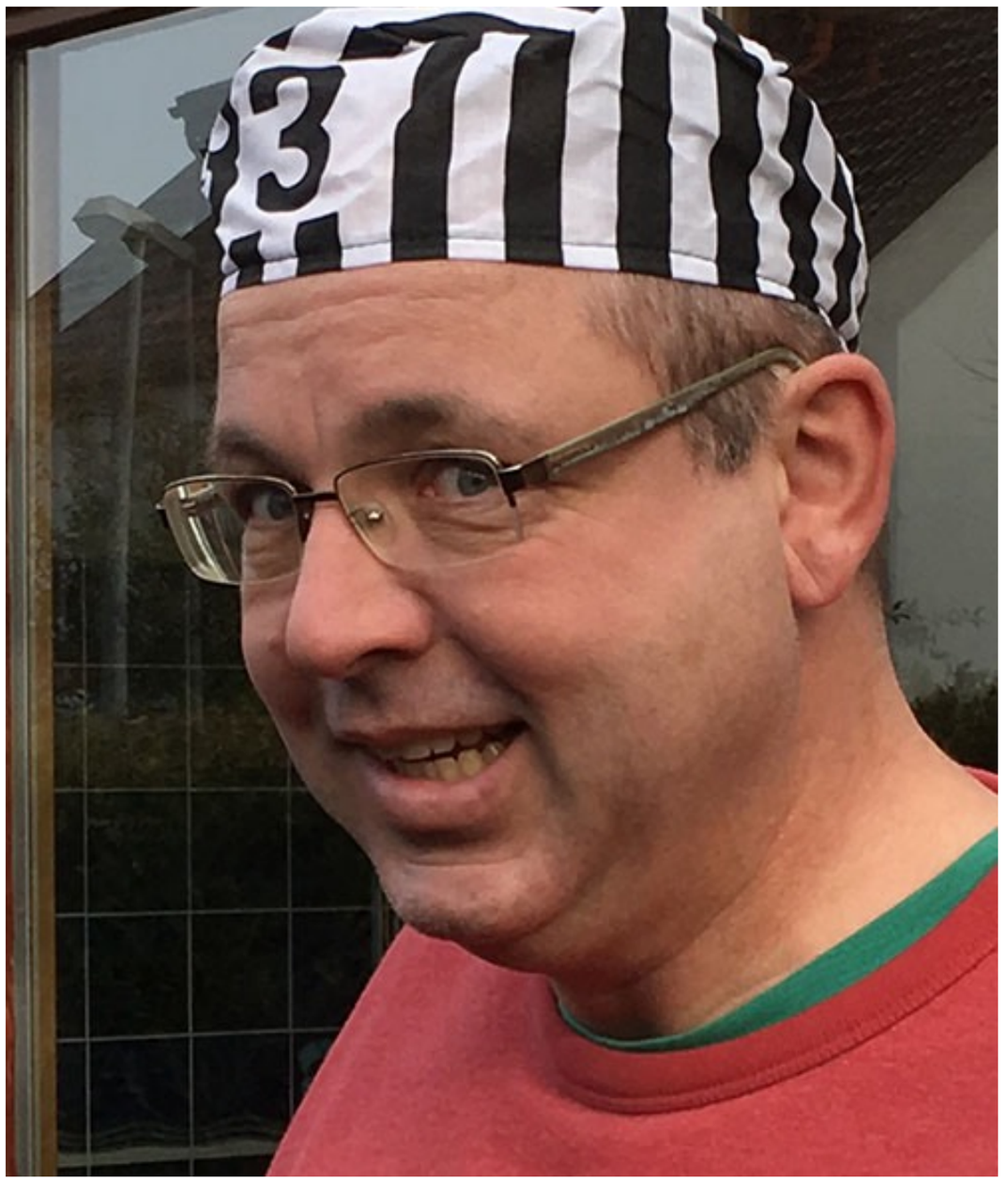
Cookies how I use them

- The user's web-browser get a cookie, this is a variable with some content in it

- reader_cookie_id = 14

- Variable name „reader_cookie_id“ << always the same

- 14 << for every webbrowser a different number



Cookies, what I do not do

- Cookie-**Pinning** to a client's ip address
  - because mobile IP changes when user is moving
  - because IPv6 changes frequently
  - an enterprise-proxy serves many users (Siemens, cable provides with not enough ipv4-addresses)

A view behind the curtain

- Managing readers (of a public library) in the web-application in a database

reader_cookie_id	status	reader_passport_nr
14	unknown	-1
15	reader	15006
…		

A view behind the curtain

A. Server gets no cookie from user -> create new cookie in the database and send that with the server's answer to the user (problem: many write-access to the database, many many cookies for users who do not accept cookies)

B. Server gets no cookie from user, wrong name or already invalid because of time -> A

The fear of the server

- What if the user know how the system works and manipulates it?

- User gets cookie-value 14 and sends 15 back with the next request -> change his identity (see internal database table 2 slides before)

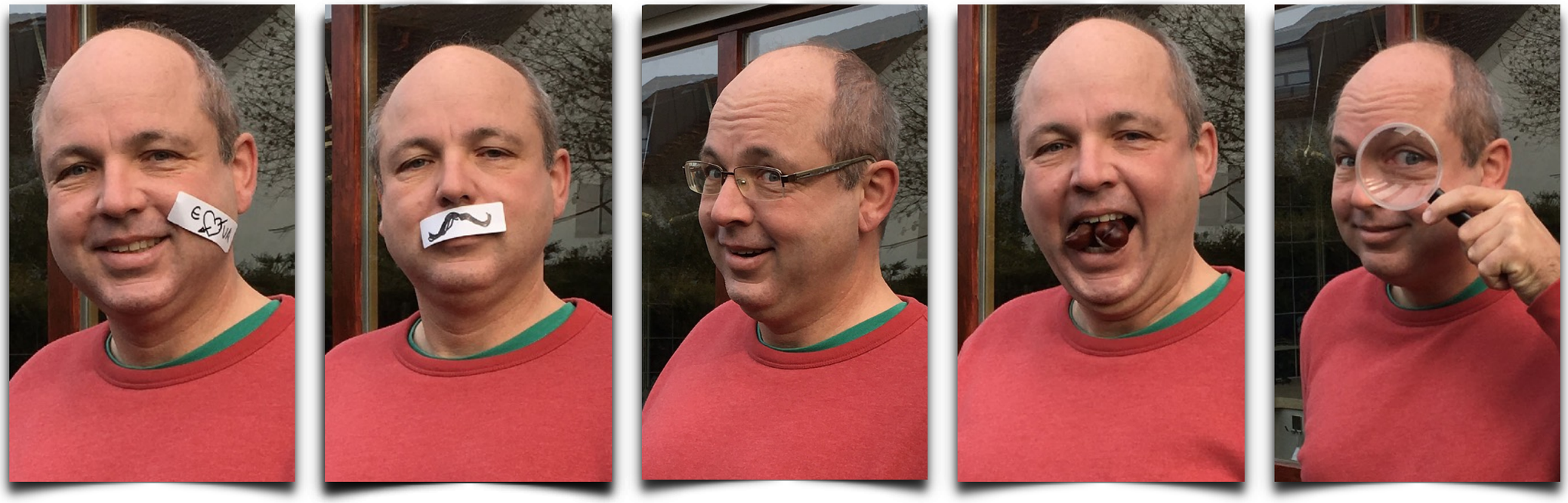
- solution: a (1) **very big** (2) **random** number as value for the cookie -> a hash value

What is a hash-value?

- A hash value is a very big number, for example MD5-hex
- 28 fe 14 b3 00 46 22 28 00 01 46 … 28 46
- The nice things about a hash value

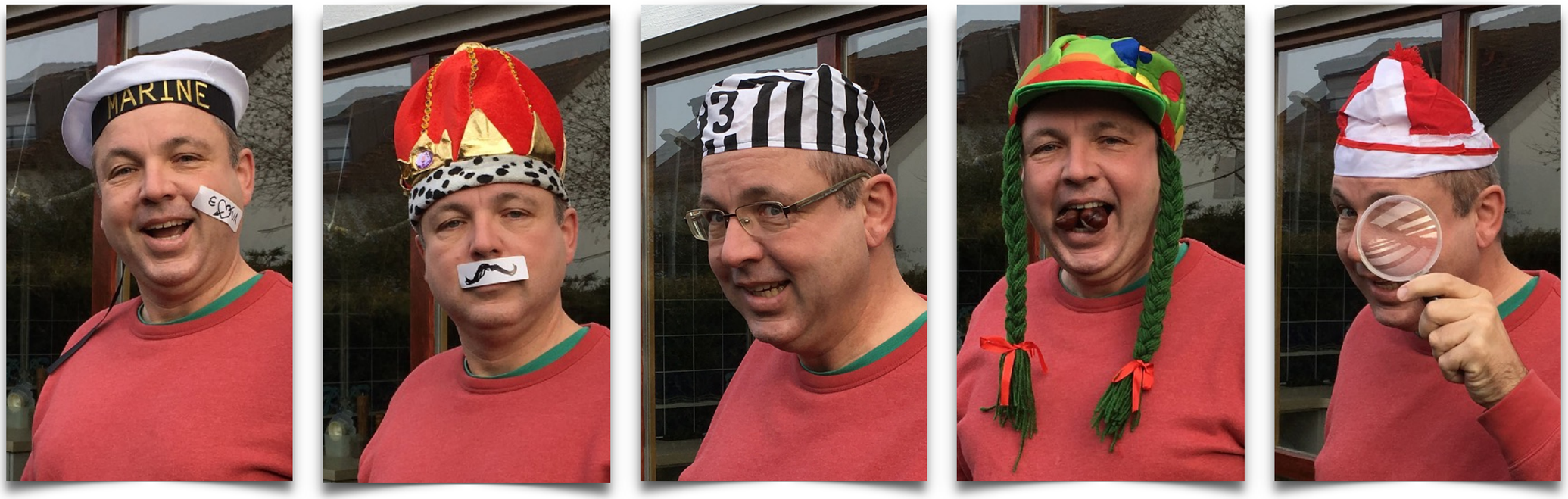
1. always the same length (32 characters for md5/hex), nice to save in a database
2. defined characters (0-9, a-f)
3. little changes in the input value create completely different hash-values
4. „relative“ collision-free (do I trust the hash-algorithm? rc4, md5, sha256, …)

When all is done well the server knows every user by it’s „hash“-hat easily



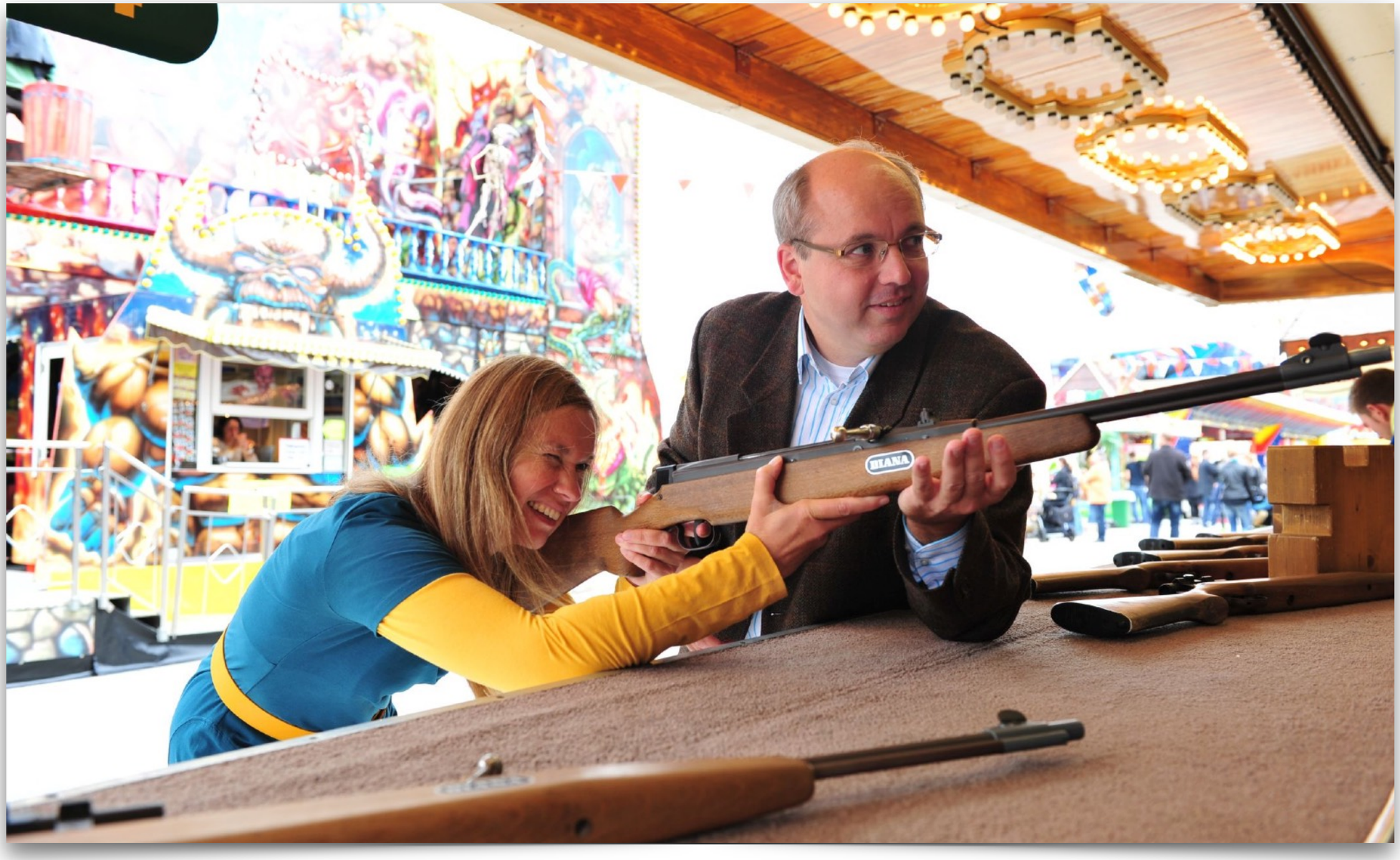
Perfect!

But how do we create different hashes for each user?



PID as hash-seed

- in my development environment there is a new process id (PID) for every request. PID is always different
- 1004 -> ab:cd:12:34:…
- 1005 -> 45:67:90:12:…
- 1006 -> 94:3f:5b:29:…

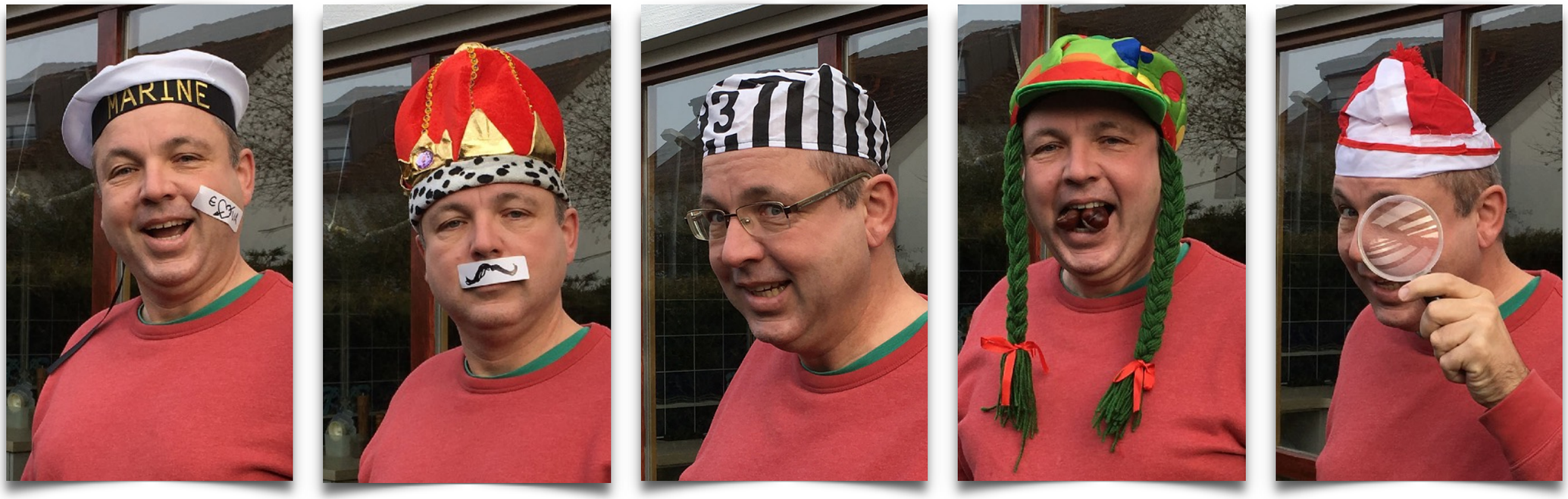


Checking reality #1

PID as hash-seed - **development** environment

Perfect!

Every user gets a different cookie-hash because PID is always different

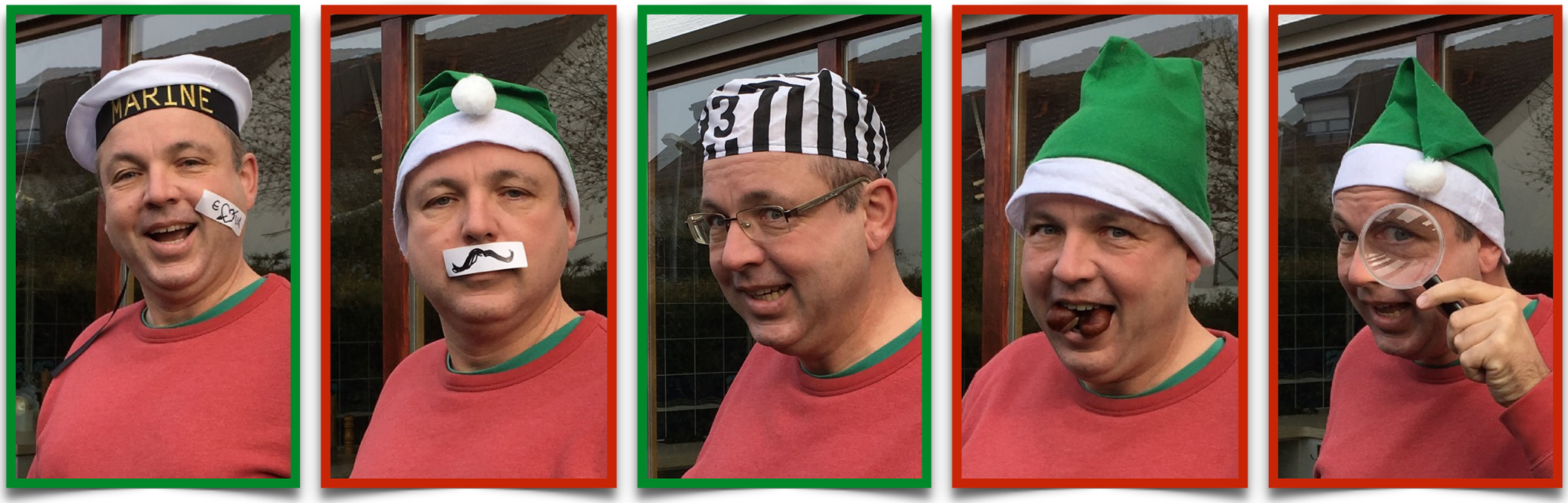


Checking reality #2

PID as hash-seed - **production** environment

NOT Perfect!

Long-living processes give their long-living PID as cookie-hash to the user



Problem:
PID as hash-seed

- Problem: there are only three long-living Fast-CGI-Processes in production environment which are having their PID for a long to (decreasing startup-time-lag)

- Problem: there are only 65.536 PIDs, then the operating system starts from „0“ again.

- The seed of the hash is not “random” enough

„A hash is a recipe to calculate something. a defined input gives always the same output.“

Time as hash-seed

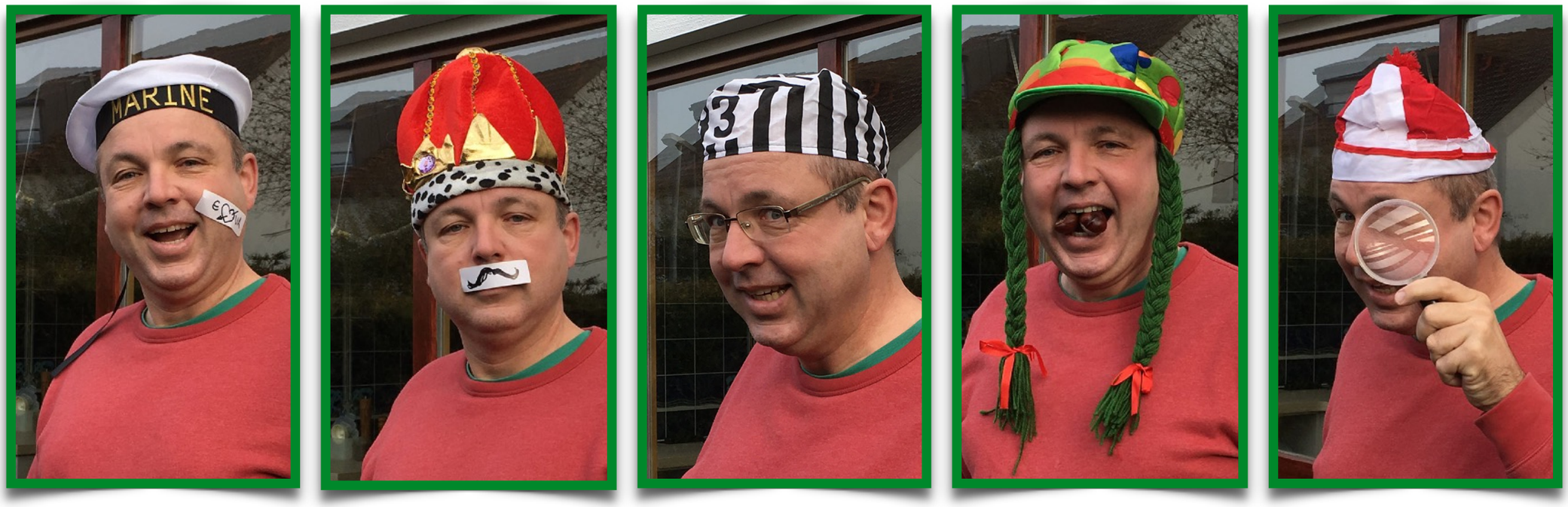
- How can we habe an input which is more different all the time?
- Time! 2016-12-31 14:18:32 << changes every second
- Hash: a small change in the seed changes the hash-output very much:
  - 2016-12-31 14:18:**32** -> ab:cd:…
  - 2016-12-31 14:18:**33** -> 45:12:…

Checking reality #3

Time as hash-seed - a user from time to time

Perfect!

Every user gets a different cookie-hash because time is always different

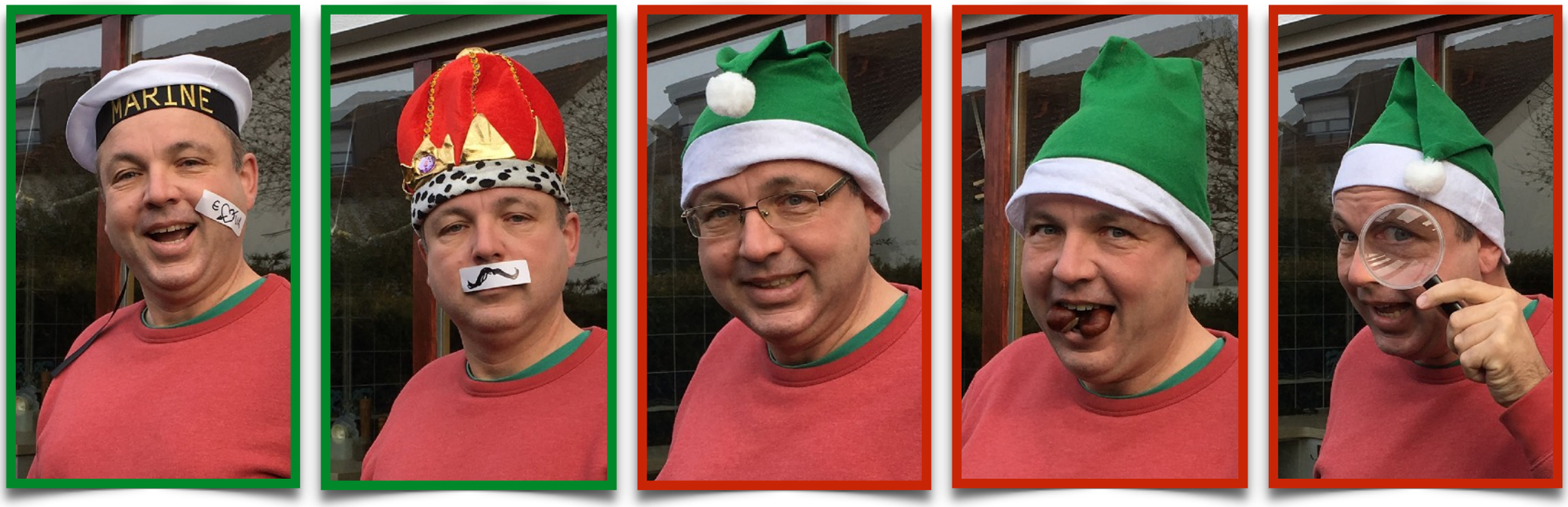


Checking reality #4

Time as hash-seed - clumps of users

NOT Perfect!

Long-living processes give their long-living PID as cookie-hash to the user



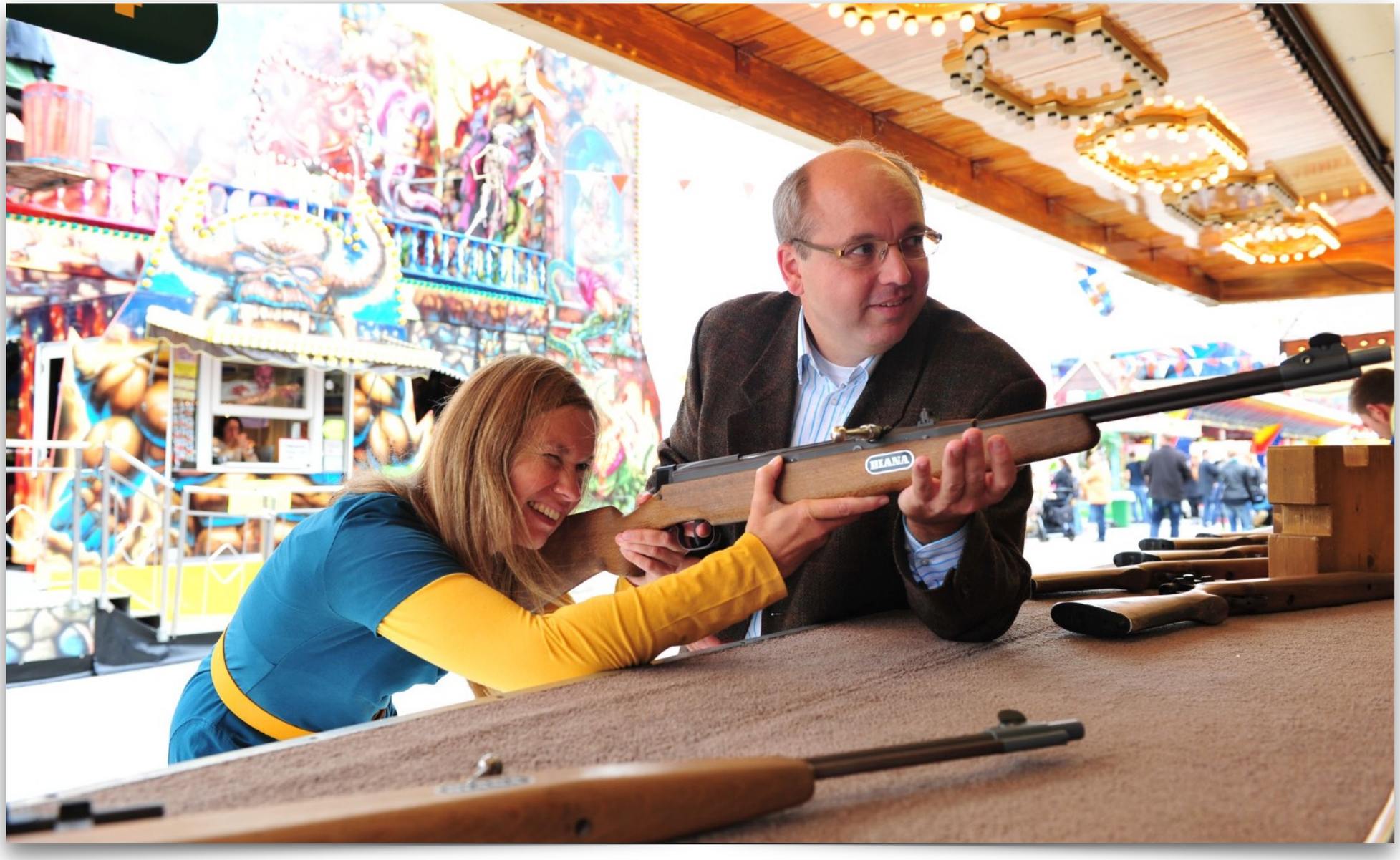
Problems:
time as hash-seed

- Time is not changing fast enough. A resolution in seconds is too rough
- With a big quantity of users (a whole country, world-wide) also a finer granularity is not enough
- There is a time which repeats: changing local time from summer-time to winter-time in Europe. Can you test this?
  - 30.10.2016 2:00 … summer-time
  - 30.10.2016 2:01
  - …
  - 30.10.2016 2:59
  - 30.10.2016 2:00 !!!! winter-time, uh, oh, local time repeats!!!!

A counter as hash-seed

- Can I have something that changes continuously?
- A persistenter counter, for example a number in a database-table cookie_seed

number(int)
14600

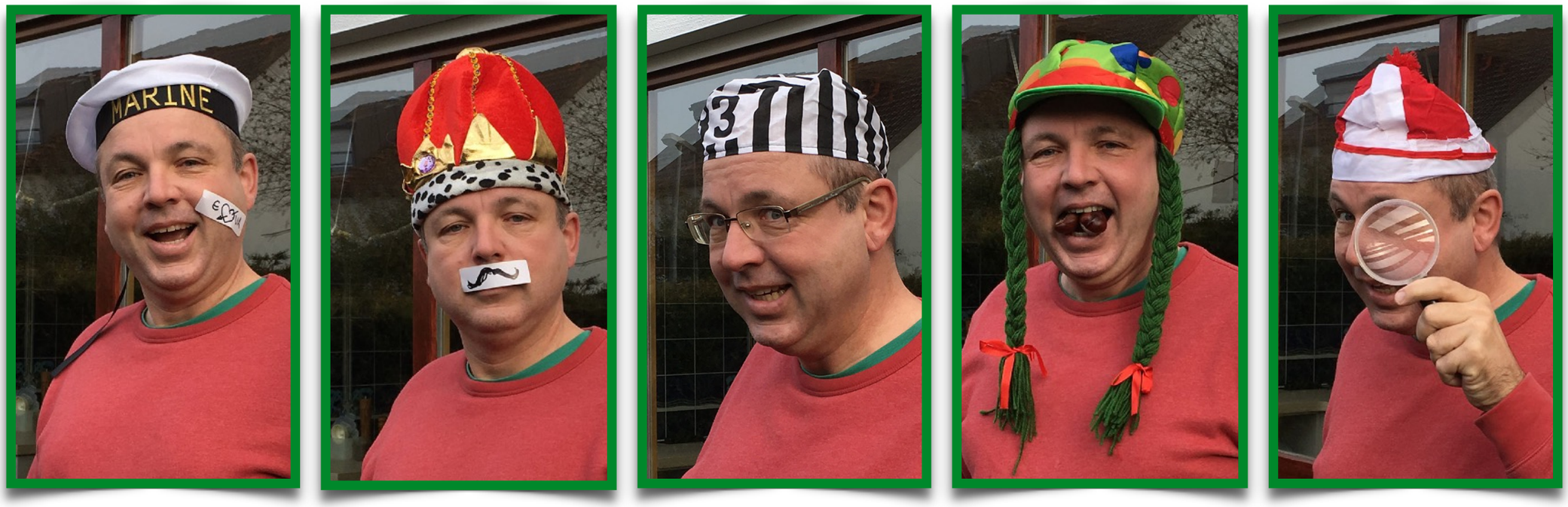


Checking reality #5

counter as hash-seed - little number

Perfect!

Every user gets
a different cookie-hash
because counter always increases

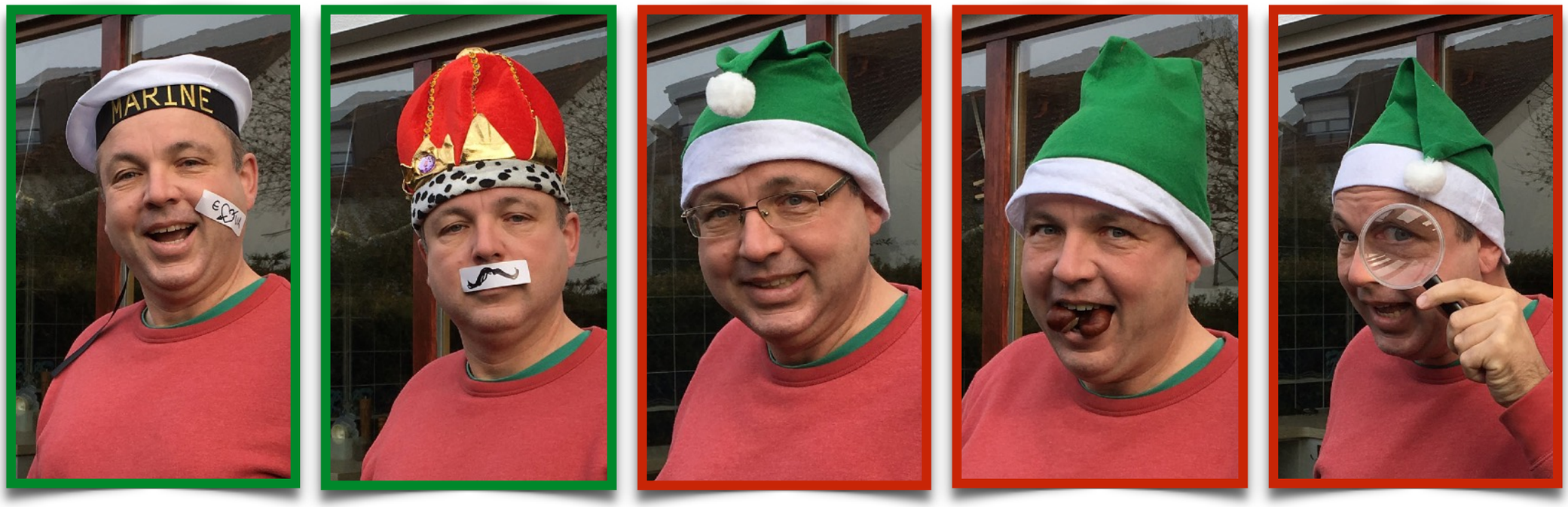


Checking reality #6

counter as hash-seed - max int

NOT Perfect!

Counter does not increase when reaching max(int)



A counter as hash-seed

number(int)
2.147.483.646

- update cookie_seed set number=number+1
- select number from cookie_seed
- gives 2.147.483.647
- increase -> gives 2.147.483.647
- increase -> gives 2.147.483.647
- increase -> gives 2.147.483.647

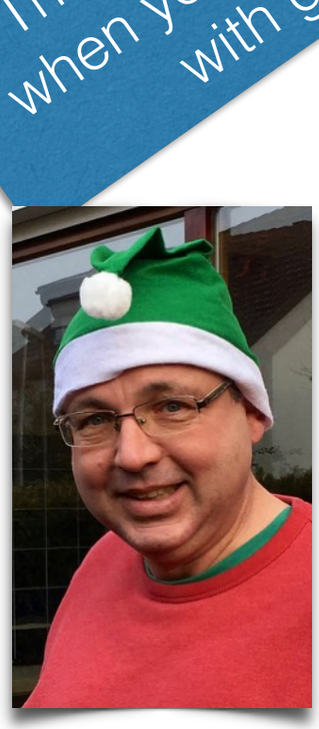
Problems:
counter hash-seed

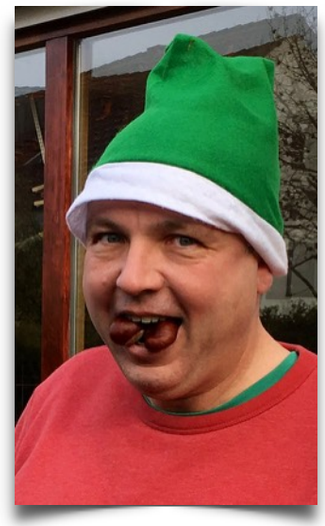
- All numbers have a maximum value on your computer depending on processor, operating system, database. You have to find out what happens when you reach that number

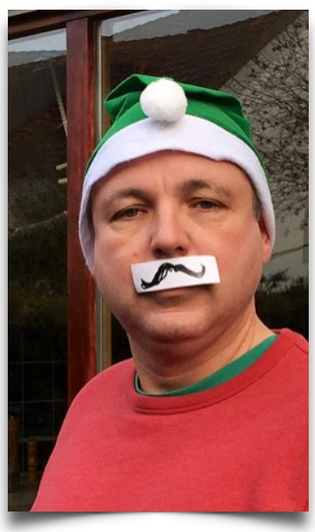
Big numbers as safety belt in cookies

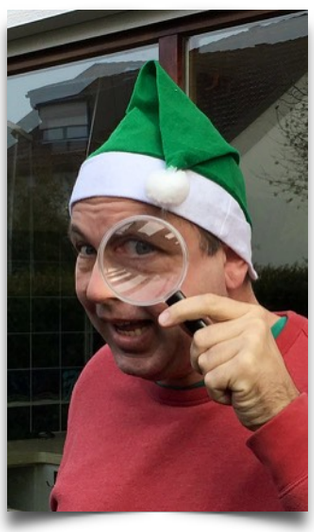
Richard Lippmann
richard.lippmann at posteo.de

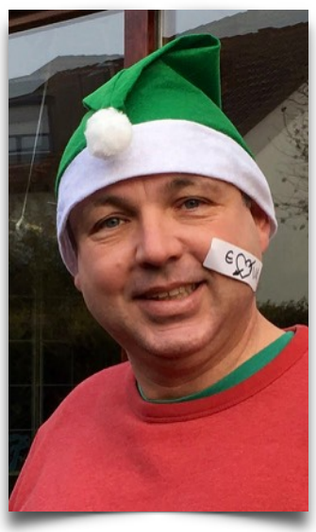
There is something wrong when you see a lot of dwarfs with green caps!











for KNF-Kongress, Nuernberg, Germany 11/2016