browser.crypto.

does it matter?

browser.crypto.

nope

javascript cryptography

considered harmful

browser.crypto.

http://www.matasano.com/articles/javascript-cryptography/

let's see why it's doomed

js served over http

man-in-the-middle attack

client <----------  ----------> server

serve different files

change files in transit

inject <script> tags

do whatever

alright, let's fix it

browser.crypto.

https

~~man-in-the-middle attack~~

client <----------> server

https

mission accomplished ?

man-on-the-server attack

client <----------> server

works with https, too

client <----------> server

https

punchline:

browser crypto

does not

change our trust model

we still need to trust the server

to do the right thing

if you don't trust the server to do

crypto for you

how could you trust its js crypto code?

„browser crypto changes nothing,
it only offers additional attack surface
- so why bother with it?“

note:

not a javascript problem per se

tho there are javascript problems as well

many algorithms require

exact-width integer operations

(e.g. on 8 bit, 32 or 64 bit)

no out-of-the-box support

for binary data

no out-of-the-box support

for big integers

workarounds lack native support

-> workarounds are slow

then: browser js problems

lack of a universally supported

„cryptographically secure
pseudo-random
number generator“

(aka csprng)

Math.random is predictable

csprng is at the ♥ of crypto

high-profile target

without it or when broken:

crypto === lulzcrypto

workarounds using mouse movement

and whatnot

meh

clearly something that should be built-in

window.crypto.getRandomValues

browser.crypto.

lacks wide-spread support

what makes javascript fun

is also what makes javascript crypto hard

dynamic runtime environment

browser.crypto.

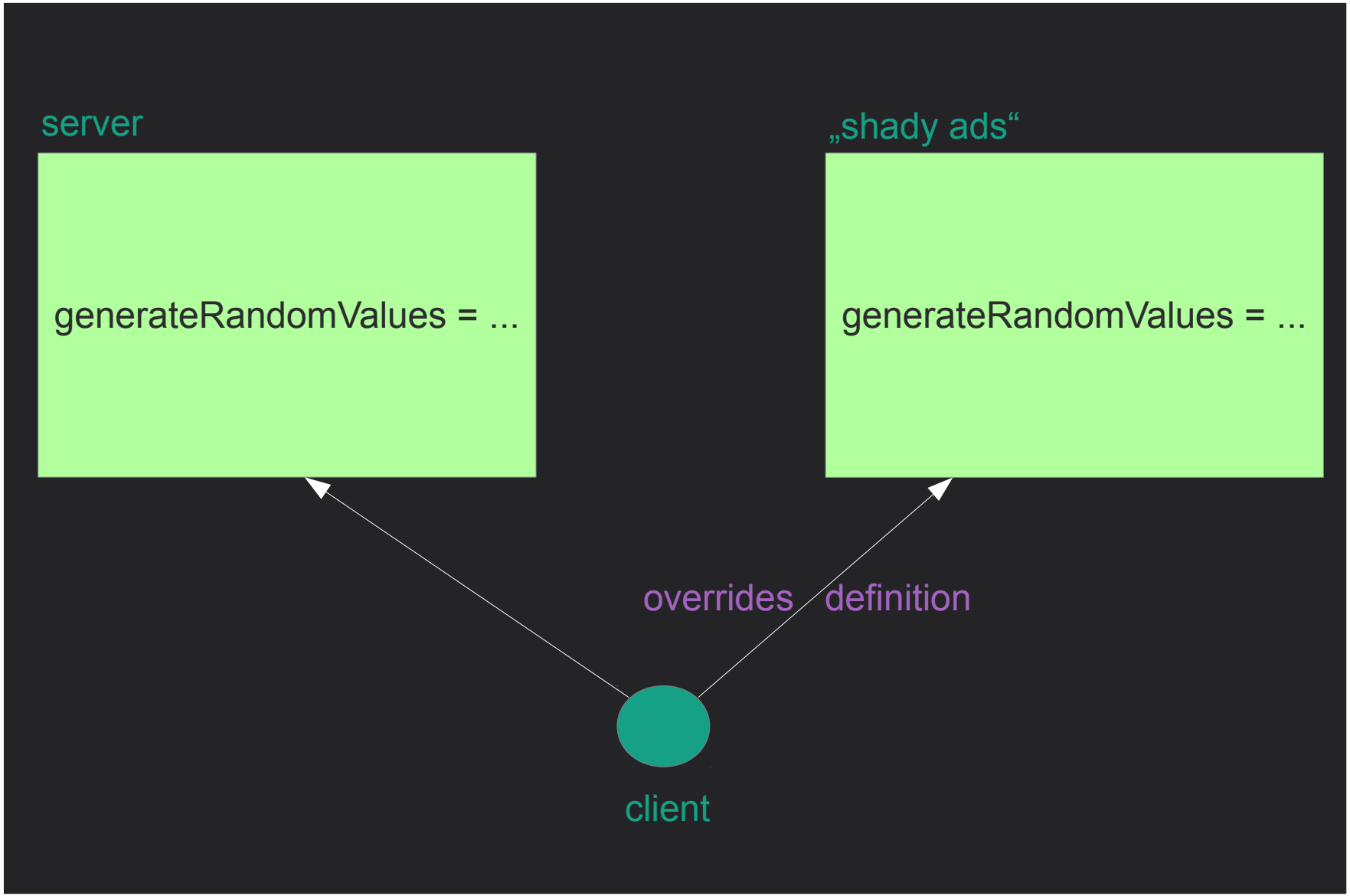
```
crypto.generateRandomValues = function(array) {
   array[0] = 42;
};
```

doesn't even have to be intentional

browser.crypto.

```
/**
 * FTFY. It's web scale now.
 */
crypto.encrypt = function(key, value) {
   /* military-grade ROT26 algorithm */
   return value;
};
```

browser.crypto.



environment could be changing continuously

„javascript crypto sucks -
have fun at realtimeconf!“

browser.crypto.

why it matters!

it's harmful alright...

but only when it's harmful

it doesn't have to be

if browsers give us the right tools

browser.crypto.

widely-adopted
browser-built-in

crypto functionality

asm.js

exact-width integer types

typed array support

(http://www.khronos.org/registry/typedarray/specs/latest/)

w3c web cryptography api

biginteger support

built-in csprng

lots of good stuff

content security policy

sandboxes

immutability of core features

to control runtime environment

„wait a second!
even if we had that stuff -
didn't you say that
nothing would change
anyway?“

i respectfully dissent

privacy

„if you have

nothing to hide,

you have nothing to fear.“

bullshit

today's surveillance is like radiation

privacy matters

(even if you don't think it does)

„what can we as individuals
do to protect our data?“

encrypt shit

browser.crypto.

on the client

-> gonna need browser crypto

„so what – i really
have nothing to hide.
just let them have it.“

surveillance

will change our lives

innocent until proven guilty

becomes

guilty until proven innocent

determinism

automated profiling

we are more than
a physical appearance
plus
the sum of our actions

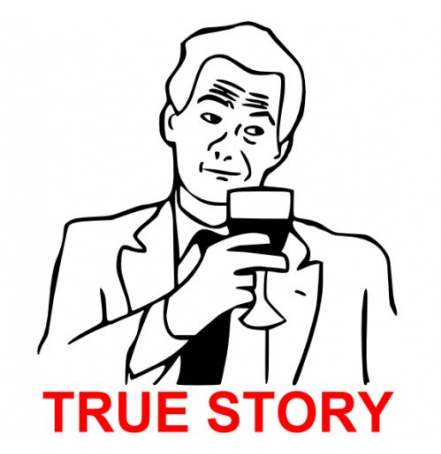
„ok, cool story, bro,
but I want some examples!“

example #1

passwords

how it's done today

the facepalm way



i forgot my password for my favorite local cinema.

they sent it to me in an email - in plain text.

the ok way

password is sent via https

server takes plaintext password

and stores a pbdkf2/bcrypt/scrypt hash

the problem with this?

password is sent via https

server takes plaintext password

and stores a pbdkf2/bcrypt/scrypt hash

i don't want to give the server
my password!

browser crypto to the rescue

pbdkf2/bcrypt/scrypt hash is sent via https

server takes hash

and stores it

but they could just send a malicious js file...

yes,
but think of admins etc.

example #2

boxdrop

any resemblance to actual applications
is purely coincidental

assume boxdrop is a service that

stores files that are uploaded

file is uploaded in plaintext

boxdrop stores the file in plaintext

the problem with this?

file is uploaded in plaintext

boxdrop stores the file in plaintext

i don't want boxdrop to see
my nekkid pictures!

sure, i can encrypt them locally

i can also use

pgp,
s/mime,
client-side encryption tools

in general, no doubt

but how often did my mom use them lately?

opt-in ain't gonna work

what we need is implicit

security by default

example #3

html5 apps

html5 – the mobile „native app“ of the future?

i hope so

because

„no, random news site,
i don't want to download your
fucking 50mb app!!!“

either way, html5 apps

will need ways to ensure

privacy/integrity/authenticity

-> browser crypto

example #4

european eid cards

great in theory

„mortimer, i just got a call
from the 90s – they said
they finally wanted their
java applets back.“

a whole industry just realized

that their products are useless

have a ♥ for europe

please, browser people:

pkcs#11 support

strong authentication
document signatures
...
-> lots of $ to be made

and, not to forget, more secure apps

example #5, #6, #7, ...

secure messaging
credit card numbers
authentication
...

final verdict

javascript crypto considered
to have a bright future

w3c api is a major step forward

it doesn't change the trust situation, but:

extortion will be useless

(good for companies)

private data is private

(good for us)

apps can no longer access secrets

and we are given the tools
to make javascript crypto
less harmful

thank you!